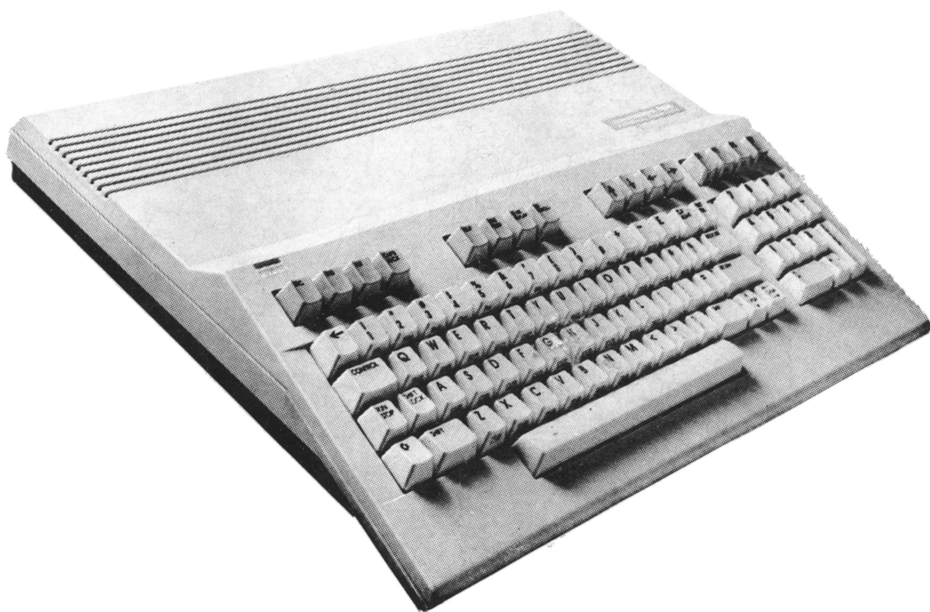
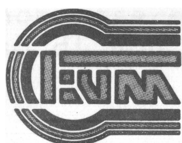
Guida di riferimento al C128





EVM-COMPUTER

Guida di riferimento al commodore 128



EVM·COMPUTER

I Edizione Ottobre 1985

EVM Computers Srl
Via Marconi 9/a

52025 MONTEVARCHI (AR)

INTRODUZIONE

Sicuramente il C128 e' uno dei migliori prodotti presenti sul mercato degli HOME COMPUTER. Cio' sopratutto per quanto riguarda il rapporto prezzo/prestazioni, il numero di periferiche collegabili, le possibilita' applicative.
Cosa importantissima non va dimenticato che relativamente ai modi di funzionamento CBM64 e CPM esistono sul mercato una mole impressionante di programmi e molti ne verranno sviluppati per il modo C128.
Questo GUIDA DI RIFERIMENTO al C128 e' stato scritto per fornire uno strumento di lavoro preliminare, anche se non per questo generica, per una parte delle capacita' operative del computer a tutti coloro che intendano iniziare a lavorarci sopra siano essi programmatori professionisti che potranno trovare indispensabili riferimenti o hobbisti che si avvicinano magari per la prima volta ad un computer.
Cerchiamo di essere infatti subito chiari.
Le capacita' del C128, che possiamo definire 3 computers in un solo contenitore, sono numerose al punto tale che non possono, almeno a nostro modesto parere, essere contenute in un unico volume.
Dei modi operativi disponibili, cioe' CBM 64, C128 e CPM, abbiamo deciso di trattare approfonditamente solo il modo 128.
Per il funzionamento in modo CPM stiamo preparando un apposito volume e non poteva essere diversamente data la mole di dati ed informazioni da trattare.
Per non essere eccessivamente ridondanti abbiamo deciso di non fare che alcuni riferimenti al

modo di utilizzo CBM64, perche' alla fine ne sarebbe risultato un manuale troppo lungo, complesso e costoso.
Abbiamo preferito venire incontro a coloro che desiderano approfondire con una speciale offerta che e' contenuta al termine di questo volume e che permette di acquistare la GUIDA AL CBM 64 ed altri prodotti a condizioni particolarmente vantaggiose.
Questo manuale inoltre, ci auguriamo almeno, possa essere il primo di una serie per il C128.
Gli altri tratteranno del Sistema Operativo, della Programmazione, delle Periferiche collegabili, dell'Hardware ecc...
Anzi possiamo subito annunciare che i primi volumi ad essere pubblicati saranno quelli relativi alla periferica disco e probabilmente al CPM.
La prima parte del volume tratta i comandi e le funzioni disponibili del C128 in ordine alfabetico con una descrizione accurata e con esempi applicativi.
Questa parte si e' resa necessaria per chi desidera avere in italiano queste informazioni, avendo magari solo il manuale in inglese, o in tedesco, oppure per chi vuole fermarsi solo al BASIC 7.0 di questo computer.
Non si parla invece di programmazione tranne che negli esempi perche', sempre a nostro parere, l'argomento e' molto complesso e richiede una trattazione particolare, appunto un manuale di PROGRAMMAZIONE.
Si parla invece di programmazione nella parte successiva del volume.
Subito dopo i comandi e le funzioni si incomincia infatti a trattare delle notevoli capacita' grafiche che il C128 mette a disposizione.
I comandi implementati per la Grafica sono infatti cosi' potenti che permettono di operare

sugli SPRITES e sulla grafica in generale in modo piu' accurato e veloce di quanto avviene per altri computers o anche sullo stesso modo CBM64, dove siamo costretti ad usare una infinita' di comandi PEEK e POKE.
Stesso discorso vale per la parte dedicata ai suoni. In questo caso forse sarebbe stato necessario un approfondimento della parte musicale che pero' esenta dalle finalita' proprie dell'opera.
La sezione relativa alla gestione delle periferiche e' stata in gran parte tratta dal volume LE PERIFERICHE COMMODORE Ed. EVM a cui rimandiamo per approfondimenti anche HARDWARE.
La parte maggiore di questi capitoli e' dedicata al registratore a cassetta per due motivi.
Prima di tutto perche' e' la periferica piu' diffusa sugli HOME COMPUTERS anche se chi intende poi lavorarci seriamente passa all'unita' a dischi.
Secondo motivo perche' e', relativamente, piu' facile da spiegare e comunque costituisce un buon inizio per operazioni sui Files.
Per sfruttare in modo notevole, ma sopratutto completo le possibilita' applicative delle periferiche, compreso le stampanti e le varie uscite e porte di connessione rimandiamo comunque al citato manuale e al volume sul CPM di futura edizione.
Abbiamo poi dedicato la penultima parte al Linguaggio macchina ed ad un'illustrazione del complesso Sistema Operativo ilcui manuale sara' pubblicato quanto prima.
Al termine abbiamo riportato, assieme ad una serie di tavole che risulteranno di grande utilita' pratica al programmatore, anche la gestione non generica degli errori e alcuni programmi esemplificativi.
Ci auguriamo di aver almeno in parte soddisfatto le aspettative dei lettori.

Restiamo comunque a loro disposizione come sempre per ogni chiarimento che possano desiderare.
Invitiamo inoltre i lettori a segnalarci tutti gli errori che inevitabilmente saranno presenti in questa edizione.
Ricordiamo che coloro che ci rinvieranno l' apposita cartolina al termine del manuale saranno inseriti nella EVM MAILING LIST e potranno essere periodicamente tenuti aggiornati di tutte le pubblicazioni, programmi, progetti HARDWARE che verranno messi a punto dalla ditta che ha curato la pubblicazione di questo manuale.

NOTA FINALE

CBM64, VIC20, C128 sono marchi registrati della COMMODORE INTERNATIONAL

CPM e' un marchio registrato della DIGITAL RESEARCH.

CAPITOLO PRIMO

Prima di iniziare la trattazione dei singoli comandi e delle funzioni del C128 vediamo alcuni concetti di carattere generale.

INIZIALIZZAZIONE

All' accensione del sistema viene generato un reset generale e vengono inizializati sia il Sistema Operativo che l' interprete Basic.
In rapporto al tipo di computer utilizzato si puo' avere una inizializzazione anche con un comando SYS diretto o sotto controllo di programma.
Prima pero' di accendere il computer ricordiamo che dei tre modi operativi disponibili viene scelto subito il modo C128 a meno che non sia collegata l' unita' a dischi e su questa sia presente il dischetto di CPM perche' in questo caso eseguira' il BOOT, cioe' l'avviamento di questo sistema operativo.
Altra cosa da osservare e' se il tasto relativo alla selezione colonne (40/80) e' posto in accordo con il tipo di monitor che si sta usando.
In altre parole se questo tasto e' premuto indichiamo al sistema che si desidera usare un monitor a 80 colonne. Se invece il tasto e' in posizione normale operiamo con un monitor a 40 colonne.
Per le varie configurazioni e le possibilita' di selezione vedi l' apposita tavola in fondo al volume.

MODI OPERATIVI

Ricordiamo che per quanto molte delle informazioni siano applicabili a tutti i modi di funzionamento intendiamo riferirci come detto all' inizio solo al modo C128.
In modo diretto, i comandi e le funzioni Basic non sono preceduti da numeri di linea e vengono eseguiti cosi' come sono stati immessi.
In questo modo i risultati di operazioni aritmetiche e logiche possono essere immediatamente visualizzati e immagazzinati per usarli piu' tardi, ma le istruzioni che hanno portato a questi risultati sono perse dopo l' esecuzione.
Questo modo di operare e' utile per lo sviluppo ed il controllo e per usare il computer come una calcolatrice sebbene molto potente.
Il modo indiretto o MODO PROGRAMMA e' invece il metodo usato per inserire programmi.
Le linee di programma sono preceduti da numeri di linea ed i comandi in esse contenuti vengono invece immagazzinate nella memoria centrale.
Successivamente il programma puo' essere fatto GIRARE con un semplice comando di RUN.

FORMATO DELLA LINEA

Le linee di programma in Basic hanno il seguente FORMATOo:

nnnn Comando Basic:Comando Basic: ecc.

Dove:

nnnn sta ad indicare il numero di linea o di

riga del programma

Come si vede nella formula possono eserci piu' comandi Basic in una sola linea di programma che dovranno essere separati solo dal carattere DUE PUNTI (:).

Una linea di programma Basic deve naturalmente incominciare sempre con un numero di linea, contenere un massimo di 80 caratteri, e vedremo poi come, e terminare con un RETURN, cioe' un ritorno carrello.

NUMERI DI LINEA

Come abbiamo detto ogni riga di programma inizia con un numero di linea. I numeri di linea che controllano l' esecuzione SEQUENZIALE del programma indicano anche l' ordine in cui le linee dello stesso programma sono immagazzinate in memoria e sono anche utilizzate come punti di riferimento per i salti di programma e per l' EDITING.
I numeri di linea devono essere compresi nell' intervallo fra 0 e 63999 e devono essere sempre numeri interi.

LA PROGRAMMAZIONE

Vediamo ora, pur nei limiti fissati nell' introduzione, di accennare alla programmazione.
Con l'evolversi delle tecnologie sono stati modificati certi concetti operativi il che ha permesso di affrontare i problemi della

programmazione sotto un' ottica diversa rendendo piu' accessibile e sopratutto piu' funzionale l' avvicinamento all' informatica.
Il compito essenziale del programmatore e' rimasto comunque quello di esprimere sotto forma di algoritmo il problema in esame e tradurlo in linguaggio, in questo caso il BASIC, sotto forma di programma.
Il concetto di algoritmo, che in matematica puo' definirsi come:

"Quell' insieme di regole atte a definire un procedimento di calcolo al fine di ottenere un determinato risultato partendo da determinati dati iniziali"

Assume in informatica un significato piu' specifico. Vediamolo.

ALGORITMO INFORMATICO

In informatica l' algoritmo deve soddisfare le seguenti caratteristiche:

1)Essere finito e terminare dopo un numero determinato di operazioni.

2)Essere definito e preciso. Ogni istruzione deve essere definita incontestabilmente.

3)Essere precisato il campo di applicazione dei dati di entrata.

4)Possedere almeno un dato in uscita.

5)Essere possibile. Devono cioe' poter essere effettuate tutte le operazioni in modo esatto e finito nel tempo, da un uomo che impieghi i mezzi manuali.

Enunciato il problema, il primo compito del programmatore e' quello di costruire un algoritmo che soddisfi le caratteristiche elencate, procedendo quindi alla stesura di un programma che conduca ad un risultato.
Lo schema sara' quindi:

PROBLEMA

ALGORITMO

PROGRAMMA

RISULTATO

Teniamo presente che:

A)-Non per tutti i problemi e' possibile costruire un algoritmo, ma cio' non significa che il problema non abbia una soluzione.

B)-L' algoritmo che risolve il problema puo' non essere unico.

C)-Il fatto che esista una soluzione per un problema, in uno o piu' casi particolari, non significa che il problema sia sempre risolubile.

D)-Il fatto che un problema non sia risolubile, non significa che non ha soluzioni, ma solo che non esiste o non siamo in grado di costruire, un algoritmo capace di dare una soluzione in

tutti i casi in cui essa esista.

Tradizionalmente, in informatica, viene usato un sistema grafico per la stesura dell' algoritmo. Questo grafico che prende il nome di schema a blocchi o FLOW CHART, consiste nella descrizione delle diverse operazioni, in forma di schema simbolico descrivente i diversi ordini e le diverse condizioni trattate dall' algoritmo stesso.
L' utilizzo di linguaggi come il Basic, anche se non completamente strutturati, o almeno spesso utilizzati in modo non strutturato, ha ridotto notevolmente l' importanza dello schema a blocchi, riservando l' uso per casi complessi.
Molto spesso, infatti, e' sufficiente scrivere con esattezza cio' che si vuol fare, per riuscire a stendere il programma.
Visto poi che il Basic e' un linguaggio interprete, il programma, se si dispone di un calcolatore, puo' venire direttamente inserito e controllato passo passo.
E' buon sistema comunque, studiare a fondo il problema e stendere una traccia scritta di cio' che si vuol fare, prima di passare alla stesura del programma definitivo per il C128
Cio' perche', in caso di errori, la macchina segnala solo quelli sintattici o di procedura e non quelli di logica.
Questi ultimi devono essere individuati dal programmatore, analizzando i risultati eventualmente ottenuti.
Le situazioni in cui ci si imbatte programmando sono riassumibili in tre punti essenziali:

I) SEQUENZA

Le operazioni devono svolgersi una dopo l' altra, secondo uno schema preordinato.

II) DIRAMAZIONE

Si deve operare una scelta in funzione del risultato ottenuto.

III) INTERAZIONE (ciclo)

Si deve ripetere un determinato numero di volte una sequenza di operazioni.

I concetti qui esposti rappresentano una sintetizzazione di quelli che sono i problemi e le tecniche della programmazione.
Lo studio del Basic, almeno come e' riportato in questo manuale, servira' a farvi entrare in possesso degli strumenti idonei a risolvere il vostro problema.

LE RIGHE DI UN PROGRAMMA

Utilizzando delle istruzioni PRINT dirette o altro e' possibile far si che il computer risponda direttamente alle nostre domande.
Cioe' il Basic esegue immediatamente il comando dato dopo che e' stato effettuato un ritorno carrello.
In questa maniera, il C128 viene usato come un supercalcolatore.
Ad esempio se si vuole eseguire un' addizione scriveremo :

```
? 2+8
```

ed avremo come risultato:

```
10
Ready.
```

Il C128 cioe' obbedisce istantaneamente a qualsiasi istruzione assegnatagli attraverso la tastiera tranne quando e' in esecuzione un programma Basic.
Il modo diretto e' valido e quindi utilizzabile quando si vogliono correggere programmi.

Tuttavia tranne questi casi per utilizzare in pieno le capacita del computer e' necessario scrivere dei veri programmi Basic.
La differenza piu' evidente in questo modo di operare e' che con un programma parecchie istruzioni possono essere raggruppate assieme e il Basic le eseguira' in successione logica cioe' una dietro l'altra .
Le operazioni necessarie per scrivere un programma sono molto semplici.
Qualsiasi istruzione si desideri che venga trattata dal Basic del C128 come un' istruzione di programma, deve essere preceduta da un numero di linea:

nn Linea di programma

Un numero di linea puo' essere un qualsiasi numero compreso fra 0 e 63999.
Tuttavia e' buona abitudine, quando si sviluppi un programma, assegnare i numeri di linea incrementandoli di 10 .o di 100. Usare cioe' 10,20,30,40, ecc. invece di 1,2,3,4 ecc.
Questo vi permettera' di avere uno spazio fra una istruzione e l' altra in modo da poter aggiungere successivamente delle nuove linee ed effettuare agevolmente correzioni del vostro programma.

Infatti il linguaggio e' stato messo a punto in modo tale che dovendo per esempio inserire fra le linee di istruzione 10 e 20 la linea 15 sia sufficiente digitare:

15 comandi o funzioni

e questa nuova linea viene AUTOMATICAMENTE immessa fra la 10 e la 20 senza nessun bisogno di intervento manuale.
Al RUN, cioe' dopo aver dato il comando RUN, il Basic eseguira' ciascuna istruzione in ordine crescente di numero di linea (tranne i salti come vedremo in seguito).

Vediamo di seguito i comandi e le funzioni ricordando che:

1)Le parole chiave del comando o della funzione sono in caratteri doppi.

2)Viene poi descritto il formato del comando o della funzione stessa con i parametri i cui significati sono spiegati in seguito. Quando questi parametri sono fra parentesi quadre sono opzionali.

3)Le virgolette racchiudono un carattere stringa, un nome di File o una espressione.

4)Le parentesi tonde includono un parametro obbligatorio.

Ricordiamo che per separare due comandi nella stessa linea di programma basta il segno :.

APPEND

FORMATO: APPEND# (numero file),"Nome del file" [,D][(ON,)U]

FUNZIONE: Aggiunge altri dati alla fine di un file sequenziale su disco.

AZIONE: Il comando APPEND e' simile al comando DOPEN tranne per il fatto che si applica solo ai files sequenziali.
I puntatori del disco sono posizionati sotto l' attuale termine del file (EOF). I dati in aggiunta devono essere quindi scritti ed il file chiuso di nuovo.
Tutte le volte che viene usata, come nome del file, una variabile o un' espressione, queste devono essere racchiuse fra parentesi.
Nel caso non siano specificate ne' l' unita' ne' il drive, l' operazione e' indirizzata al drive 0 dell' unita' 8.

ESEMPIO:

APPEND#1,"MASTER"

Riapre il file di nome MASTER sull' unita' 8 drive 0, con un file logico 1 per le successive operazioni di PRINT#.

ESEMPIO:

APPEND#8,(B$),D0,U8

Apre il file logico il cui nome e' presente nella variabile B$ sul Drive 0 (D), periferica numero 8 e lo prepara per una funzione di aggiunta.

AUTO

FORMATO: AUTO [linea]

FUNZIONE: Abilita la numerazione automatica delle righe di un programma.

AZIONE: Se viene seguito da un valore numerico questa istruzione, che puo' essere impiegata solo in modo diretto, cioe' non puo' essere inserita in un programma, abilita il C128 alla numerazione automatica del programma BASIC che stiamo scrivendo.
In altre parole viene facilitata al programmatore la scrittura di linee di Basic perche' dopo che e' stato scelto un incremento a piacere fra una linea ed un' altra al termine di ogni linea scritta il cursore, dopo il RETURN, si posiziona a capo dopo che il sistema avra' scritto un nuovo numero di linea.
Impiegando il comando senza nessun numero di linea se ne disabilita l' uso.

ESEMPI:

AUTO 5 Numera automaticamente le linee di programma da quel punto in poi con un incremento di 5. Se siamo gia' alla linea 200 potremo cosi' scrivere la 205,210,215 e cosi' via.

AUTO 100 Numera automaticamente le linee con un intervallo di 100.

AUTO Disabilita questa funzione.

BACKUP

FORMATO: BACKUP D(x) TO D(y)

FUNZIONE: Duplica l' intero contenuto di un dischetto su un' altro dischetto.

AZIONE: L' utente specifica i numeri di drives su cui sara' posto il dischetto sorgente ed il numero del drive sul quale sara' posto il dischetto che dovra' essere la copia del primo.
Il comando BACKUP funziona infatti solo se abbiamo a disposizione due unita' a disco singolo o una unita' a disco doppio. L' operazione di copia consiste prima di tutto nella formattazione del nuovo dischetto e quindi nella copia di tutti i files incluso il formato del dischetto stesso. Poiche' si tratta di operazione distruttiva per i dati presenti eventualmente nel dischetto di destinazione il computer, prima di eseguire la copia chiede se siamo sicuri di quello che stiamo per fare con la domanda: "ARE YOU SURE?", alla quale dovremo rispondere con Y per YES.
Sara' bene ricordare di mettere la fascetta di protezione sulla tacca del dischetto originale e di lasciare libera invece quella del dischetto copia.

NOTA

Molti programmi dispongono di protezioni SOFTWARE per cui la copia con questo sistema non e' consentita.

ESEMPIO: BACKUP D0 TO D1

Copia l' intero contenuto del disco presente sull' unita' 0 su quello presente nell' unita' 1.

BANK

FORMATO: BANK numero del banco di memoria

FUNZIONE: Seleziona una dei 16 banchi di memoria del 128.

AZIONE: Normalmente il banco di memoria selezionato all' accensione e' il N. 15. Con questo comando che puo' essere utilizzato in modo programma si seleziona, cioe' si sceglie per determinate operazioni un banco diverso. La scelta di un dato banco di memoria riconfigura anche il sistema. Riportiamo di seguito una tavola contenente le configurazioni del sistema nella selezione dei banchi rimandando all' apposito capitolo sulla gestione della memoria per approfondimenti:

CONFIGURAZIONE BANCHI DI MEMORIA PER 128

BANCO	CONFIGURAZIONE
0	Solo RAM(0)
1	Solo RAM(1)
2	Solo RAM(2)
3	Solo RAM(3)
4	ROM interna,RAM(0),I/O
5	ROM interna,RAM(1),I/O
6	ROM interna,RAM(2),I/O
7	ROM interna,RAM(3),I/O
8	ROM esterna,RAM(0),I/O
9	ROM esterna,RAM(1),I/O
10	ROM esterna,RAM(2),I/O
11	ROM interna,RAM(3),I/O
12	KERNAL e ROM interna (bassa),RAM(0),I/O
13	KERNAL e ROM esterna (bassa),RAM(0),I/O
14	KERNAL e ROM BASIC,RAM(0),ROM carattere
15	KERNAL e ROM BASIC,RAM(0),I/O

ESEMPIO

```
10 BANK2
20 POKE 1126,20
```

In questo esempio abbiamo immagazinato il valore 20 nella locazione 1126 del banco di memoria 2.

BEGIN/BEND

FORMATO: IF condizione THEN BEGIN: comando

FUNZIONE: Comando di salto condizionato

AZIONE: Questo comando di salto condizionato consente di avere una struttura tale che si possano includere piu' linee di programma fra l' inizio della condizione (BEGIN) e la fine della o delle serie di condizioni END.
La cosa migliore ci sembra quella di chiarire con un esempio e poi di commentarlo.

ESEMPIO

```
10 PRINT" HO PENSATO UN NUMERO INDOVINA"
20 X=INT(RND(1)*10)
30 INPUT A
40 N=N+1
50  IFA<>XTHEN   BEGIN:   PRINTCHR$(15)"ERRORE";
CHR$(143)
60 IFA>XTHEN PRINT"TROPPO ALTO"
70 IFA<XTHEN PRINT"TROPPO BASSO"
80 BEND: GOTO 30
90 PRINTCHR$(15)"BRAVO  HAI  INDOVINATO  DOPO"N"
TENTATIVI"
```

BLOAD

FORMATO: BLOAD"nome del file"[,D numero del drive] [,U numero della periferica] [,B numero del banco di memoria] [,P indirizzo di inizio]

FUNZIONE: Carica un file di tipo binario facendolo partire da una data locazione di memoria.

AZIONE: I numerosi parametri di questo comando sono abbastanza chiari. Da sottolineare solo la possibilita' di caricare il file binario in una data zona di memoria specificata dal numero del banco e dall' indirizzo di inizio in quel banco. Ricordiamo che un FILE BINARIO e' un file che e' stato salvato attraverso un comando BSAVE, che vedremo in seguito oppure con il monitor.
Ogni comando BLOAD funziona su un SOLO banco di memoria. Per salvare informazioni da piu' di un banco si devono utilizzare piu' comandi BSAVE per salvarli in files separati. Per questo motivo, per poter utilizzare una tecnica di programmazione o comunque di utilizzo in piu' banchi di memoria, sara' necessario utilizzare piu' comandi BLOAD esattamente uno per ogni banco di informazioni. Se non si specifica il banco di memoria in cui si desidera caricare il file il sistema assegnera' per DEFAULT il banco 15 oppure il banco di memoria che e' stato selezionato per ultimo. Il valore di DEFAULT per l' indirizzo di partenza P e' invece 2.
Come vedremo queste tecniche saranno utili sia per i programmi in linguaggio macchina che per la grafica.

ESEMPI:

```
100 BLOAD "PIPPO",D0,B2,P1024
```

Con questo comando abbiamo caricato dal disco 0 il file di nome PIPPO nella memoria interna del computer a partire dall' indirizzo 1024 del banco di memoria 2.

```
100 BLOAD "FILE A",D0,U8,B1,P4096
110 BLOAD "FILE B",D0,U8,B2,P1024
```

Questo esempio mostra come caricare due files binari in memoria. Il primo file "FILE A" sara' caricato nel banco di memoria 1 a partire dall' indirizzo 4096 ed il secondo successivamente nel banco 2 a partire dall' indirizzo 1024.
Ricordiamo che se non e' specificato l' indirizzo di partenza questi sara' per DEFAULT uguale a 2.

BOOT

FORMATO: BOOT "nome del file" [,D numero drive] [(ON,)U periferica]

FUNZIONE: Carica ed esegue un programma salvato in precedenza come file binario.

AZIONE: Carica quindi un programma che e' stato salvato in precedenza non come tale, ma come file binario e ne inizia l' esecuzione ad un predefinito indirizzo di partenza che pertanto dovra' essere stato dato come parametro al momento del salvataggio del programma.

ESEMPI

BOOT Carichera' il primo programma binario che trova sul dischetto.

```
100 BOOT "PIPPO"
```

Carichera' ed eseguira' il programma binario PIPPO.

BOX

FORMATO: BOX [colore sorgente],X1,Y1 [,X2,Y2] [,[angolo][,riempimento]]

FUNZIONE: Disegna un riquadro in un punto qualsiasi dello schermo.

AZIONE: Questo comando permette di disegnare una figura a quattro lati in qualsiasi punto dello schermo. Non abbiamo voluto chiamarlo rettangolo perche' puo' essere anche un quadrato. Vediamo ora i parametri:

COLORE SORGENTE e' il colore utilizzato per il quadrilatero. Se questo parametro viene omesso verra' utilizzato il colore di Primo Piano (BACKGROUND). I valori assgnabili a questo parametro sono:

0 = Colore di Primo Piano
1 = Colore di sfondo
2 = Multicolore 1
3 = Multicolore 2

I valori 0 e 1 devono essere obbligatoriamente usati quando siamo in modo STANDARD BIT MAP mentre i valori 2 e 3 del parametro quando siamo in modo MULTICOLOR BIT MAP. Per l' impiego ed il significato vedremo dopo la sezione di questo manuale dedicato alla grafica.

X1,Y1 indicano le coordinate di arrivo del

quadrilatero.

X2,Y2 indicano, se specificate, le coordinate di partenza per il disegno oppure il PC o PIXEL CURSOR. Il PC e' simile al cursore mobile che si vede quando siamo in modo testo ed in questo caso, invece di indicare dove apparira' il prossimo carattere indica da dove inizia un disegno. Un comando che vedremo in seguito (LOCATE) ci consentira' di posizionare a piacere il PC. Quindi se non avremo specificato X2 e Y2 il disegno iniziera' a partire dalla posizione del PC.

ANGOLO questo parametro determina i gradi di rotazione in senso orario della figura intorno al suo centro. I valori possono andare da 0 a 360 gradi. Il valore di DEFAULT e' 0.

RIEMPIMENTO indica, se uguale a 1, la possibilita' di riempire o colorare l' interno della figura disegnata. Questa operazione verra' fatta utilizzando il colore scelto per disegnare il poligono. In altre parole se abbiamo scelto di disegnare il quadrilatero con il giallo, questi sara' riempito con lo stesso giallo. Il valore di DEFAULT e' 0.

In rapporto alle coordinate X e Y i rispettivi valori possono essere dati sia in assoluto che in valore relativo rispetto al PC, ed ogni coordinata e' indipendente, come assegnazione di valori, dall' altra. Vediamo la tabella delle possibili combinazioni che si possono dare per i valori assoluti e relativi ricordando che i secondi devono essere dichirati con un segno positivo (+) o negativo (-) rispetto al valore del PC.

X assoluto, Y assoluto   X,Y

X relativo, Y assoluto   +/-X,Y
X assoluto, Y relativo   X,+/-Y
X relativo, Y relativo   +/-X,+/-Y

Come potremo vedere dagli esempi riportati ogni parametro puo' venir omesso, tuttavia il suo posto, nel caso siano presenti parametri successivi, cioe' verso destra, deve essere preso da una virgola.

ESEMPI

NOTA

Ricordiamo che e' necessario entrare in modo grafico prima di provare questi esempi che altrimenti non darebbero alcun risultato visibile. Provare con GRAPHIC 2,1.

BOX 1,10,10,60,60   Disegna un quadrato usando valori assoluti.

BOX ,10,10,60,60,45,1 Ruota il quadrato e lo colora internamente.

BOX 1,0,0,319,199 Disegna un riquadro attorno allo schermo.

BOX 1,+60,+40   Disegna un poligono con punto di inizio uguale al PC e di dimensioni di 60 pixel sulla destra e 40 in basso sempre rispetto al PC.

BSAVE

FORMATO: BSAVE "nome del file" [,D numero del drive] [,U numero della periferica] [,B banco di memoria], P(indirizzo di inizio)TO P(indirizzo

di fine).

FUNZIONE: Salva un file in forma binaria da un data locazione di memoria.

AZIONE: Serve per salvare su periferica selezionabile attraverso gli usuali parametri una locazione di memoria compresa fra un indirizzo di partenza ed uno di fine in un dato banco. I valori di DEFAULT per i parametri non specificati sono i seguenti:

D = 0
U = 8
N. del Banco = 15 o l' ultimo selezionato
Indirizzo di partenza = 2
Indirizzo di fine = 65535

In altre parole se non si specificano l' indirizzo di partenza e quello della fine dela memoria da salvare verra' salvato l' intero banco scelto o di DEFAULT.
Anche con questo comando, se si desiderano salvare piu' zone di memoria e' necessario usarlo piu' volte.

NOTA

Questo comando e' esattamente equivalente al comando S che si impiega con il MONITOR. Lo vedremo utilizzato spesso nella grafica per il salvataggio di aree contenenti dei disegni.

ESEMPI

100 BSAVE "DISEGNO 1", B3, P1024 to P3094
Salva il contenuto della memoria del banco 3 da 1024 a 3094 in un file binario su disco con il nome di DISEGNO 1.

```
100 BSAVE "PIPPO 1", B0, P 1024 to P35000
110 BSAVE "PIPPO 2", B1, P 1024 to P35000
```

Con questo secondo esempio si salvano, in modo programma due zone di memoria di due banchi diversi, nei quali per esempio potevamo aver immagazzinato dei valori.

CATALOG

FORMATO: CATALOG [D numero drive] [(ON,)U numero periferica] [,stringa]

FUNZIONE: Visualizza la directory del disco.

AZIONE: Questo comando serve per visualizzare la Directory, cioe' l' indice del disco stesso.

ESEMPIO: CATALOG

Verra' visualizzata la directory del dischetto.

CHAR

FORMATO: CHAR [colore sorgente],X,Y [,stringa][,RVS]

FUNZIONE: Visualizza caratteri in una data posizione di schermo.

AZIONE: Questo comando permette di visualizzare un testo, cioe' solo lettere maiuscole, minuscole o segni grafici standard, sia che ci si trovi in modo BIT MAP o MULTI-COLOR o anche in modo di testo standard contrariamente a quanto avviene con il comando PRINT che puo'

essere utilizzato solo nel modo di TESTO STANDARD. Vediamone i parametri:

COLORE SORGENTE a questo parametro si possono assegnare i valori 0 che sara' il colore di Sfondo o 1 che sara' il valore di Primo piano. Osservare che assegnando al colore sorgente lo 0 si puo' CANCELLARE il testo visualizzato sullo schermo.

X e Y sono i parametri relativi rispettivamente alla colonna (X) ed alla riga (Y) dove il sistema deve iniziare a scrivere la stringa. I valori assegnabili sono da 0 a 79 per la colonna (ricordando che quando si lavora su 40 colonne va a capo dopo la quarantesima) e da 0 a 24 per la riga.

STRINGA e' la stringa che deve essere visualizzata. Si puo' utilizzare una stringa alfanumerica fra virgolette oppure il nome di una variabile stringa da programma (es. A$, B$, ecc.). Relativamente a questo parametro il funzionamento e' del tutto simile al comando PRINT.

RVS si riferisce al controllo relativo al carattere che sara' normale per DEFAULT o per valore 0 e in REVERSE per il valore 1.
Possono essere utilizzate le funzioni CHR$(14) e CHR$(142) per operare in maiscolo o minuscolo.
L' ultimo parametro e' abbastanza importante e deve essere visto piu' approfonditamente. Nel modo TESTO STANDARD RVS qualsiasi valore abbia viene ignorato e la stringa di CHAR viene visualizzata come quando si usa PRINT.
Nel modo ALTA RISOLUZIONE o Split SCREEN che vedremo in seguito se il parametro RVS ha il valore 0 la stringa verra' visualizzata normalmente se invece ad RVS si assegna il

valore 1 la stringa sara' visualizzata in REVERSE.

NOTA

In questi modi ed in modo MULTI-COLOR i caratteri di controllo nella stringa non sono considerati come controlli e vengono stampati.
Nel modo MULTI-COLOR gli effetti del valori assegnati al parametro RVS dipendono dai valori assegnati in precedenza al parametro COLORE SORGENTE. Se il colore sorgente e' 1,2 o 3 ed RVS=0 il testo viene visualizzato normalmente, mentre se RVS e' 1 il testo viene visualizzato in REVERSE. In entrambi i casi viene impiegato il colore sorgente selezionato.
Sempre in modo MULTI-COLOR se al colore sorgente e' assegnato il valore 0 il testo viene sempre visualizzato nel colore di primo piano anziche' nel colore di sfondo. Se anche RVS=0 i caratteri della stringa verranno visualizzati con uno sfondo di Multi-color 1. Se invece RVS=1 i caratteri verranno visualizzati con uno Sfondo di Multi-color 2.
Vediamo ora una tabella riassuntiva di quanto detto:

COLORE SORGENTE	RVS	EFFETTO
1,2,3	0	Normale nel colore selezionato
1,2,3	1	Reverse nel colore selezionato
0	0	Il testo e' nel colore Primo Piano, lo Sfondo in Multi-color 1
0	1	Il testo e' nel colore Primo Piano, lo Sfondo in Multi-color 2

Il seguente programma serve per esempio iniziale di quanto detto:

```
10  V=1:O=1:OO=1:VV=1
20  COLOR0,1:COLOR4,1:GRAPHIC2,1
30  CHAR1,O,V,"EVM",0
40  LOCATE22,1:LIST30
50  SLEEP1
60  CHAR1,O,V,"EVM",1
70  LOCATE22,1:LIST60
80  SLEEP1
90  CHAR1,O,V,"   ",0
100 O=O+OO:V=V+VV:IFO>36THENO=35:OO=OO*1
110 IFO<1THENO=1:OO=OO*-1
120 IFV>19THENV=18:VV=VV*-1
130 IFV<1THENV=2:VV=VV*-1
140 GOTO30
```

In questo esempio in cui vedrete il nome EVM apparire lentamente (comando SLEEP che si puo' anche togliere) in normale ed in reverse alternativamente so raccomanda di fare attenzione alle lettere O ed ai valori 0 (ZERO).

CIRCLE

FORMATO: CIRCLE [colore sorgente],X,Y [,Xr][,Yr] [,sa] [,ea] [,angolo] [,inc]

FUNZIONE: Serve per disegnare archi, cerchi, ellissi ed anche poligoni in una data posizione di schermo.

AZIONE: Questo comando, se utilizzato correttamente permette di disegnare una quantita' veramente notevole di figure geometriche. Per prima cosa vediamone i parametri.

COLORE SORGENTE e' il colore utilizzato per il

cerchio o per la figura che comunque si intende disegnare. Se questo parametro viene omesso verra' utilizzato il colore di Primo Piano (BACKGROUND). I valori assegnabili a questo parametro sono:

0 = Colore di Primo Piano
1 = Colore di sfondo
2 = Multicolore 1
3 = Multicolore 2

X,Y indicano le coordinate del centro della figura. Se vengono omessi viene utilizzata la posizione attuale del PC. Come abbiamo gia' visto in precedenza anche nel comando CIRCLE si possono dare ai parametri valori assoluti o relativi.

Xr questo parametro e' necessario e determina il raggio orizzontale del cerchio espresso in numero di punti.

Yr che e' invece opzionale, determina il raggio verticale della figura espresso in numero di punti. Se omesso viene impiegato il valore del parametro precedente.

SA e EA sono parametri opzionali che definiscono i punti di partenza e di arrivo di un arco di circonferenza. I valori di DEFAULT sono rispettivamente di 0 e 360. Ricordiamo che questi valori sono dati in gradi per cui 0 e' verticale, 90 a destra, 180 verso il basso, ecc.

ANGOLO indica la rotazione in gradi in senso orario. Il valore di DEFAULT e' 0, cioe' nessuna rotazione.

La posizione finale del PC sara' sulla posizione ea. A causa della differenza tra il numero di

punti orizzontali e verticali dello schermo, che ricordiamo e' di 320x200, impostando il raggio X eguale al raggio Y invece di rappresentare un cerchio rappresenteremo un' ellisse. Infatti per disegnare un cerchio dovremo creare un rapporto fra i raggi pari al rapporto fra punti orizzontali e verticali. Rapporto che dipendera' dal modo grafico che si sta utilizzando.
Il C128 disegna i cerchi come il susseguirsi di una serie di rette. Il metodo consiste nel calcolare il punto successivo sulla circonferenza del cerchio ed unirlo al precedente con una retta.

INC e' l' incremento fra i due segmenti di cui si parlava e specifica di quanti gradi (il valore di DEFAULT e' 2) il PC viene mosso al punto successivo (in senso orario). E' facile quindi comprendere che aumentando il valore assegnato ad INC il contorno del cerchio diventa sempre piu' grossolano fino ad apparire un poligono.

Vediamo ora una serie di esempi ricordando che i parametri omessi devono essere rimpiazzati con una virgola e che assumeranno i valori di DEFAULT appena detti.

ESEMPI

CIRCLE 1,160,100,100,75   Disegna un cerchio

CIRCLE 1,160,100,65;10   Disegna un' ellisse

CIRCLE,260,40,20,,,,,90   Disegna un rombo

CIRCLE 1,60,140,20,18,,,,120   Disegna un triangolo

CLOSE

FORMATO: CLOSE (numero del file)

FUNZIONE: Chiude il canale di Input/Output.

AZIONE: Il numero del file e' il numero sotto il quale il file era stato aperto con OPEN o con DOPEN.
L' associazione, cioe' il nesso logico operativo, fra un particolare file ed il numero del file, termina dopo l' esecuzione di CLOSE. Per proseguire le operazioni dopo il CLOSE, il file deve essere riaperto, (utilizzando di nuovo un comando OPEN) e potremo riadoperare sia lo stesso FILE NUMBER che un' altro.
E' importante notare che con questo comando quanto e' rimasto nel BUFFER viene memorizzato sulla periferica con la quale avevamo un canale aperto. Infatti nel BUFFER e' quasi sempre memorizzato qualcosa e se la registrazione e lo scarico conseguente al comando CLOSE non avviene il file risulta incompleto su nastro e sara' illeggibile su disco.
Se si tenta di riaprire, per errore, un file non chiuso avremo una segnalazione di errore: ?FILE OPEN ERROR

ESEMPIO:

```
10  OPEN 4,8,1, "PIPPO,S,W"
....
....
100 PRINT#4, A$,B$,C,D%
....
300 CLOSE4
```

Questo semplice esempio mostra fra l' altro l' apertura, la scrittura e la chisura del file PIPPO sull' unita' a dischi (periferica 8)

impiegando il numero di file 4.

CLR

FORMATO: CLR

FUNZIONE: Mette a zero tutte le variabili programma.

AZIONE: Questo comando, che viene automaticamente dato all' atto del RUN di un programma qualsiasi esegue le seguenti funzioni:

1)Mette a zero tutte le variabili numeriche
2)Annulla tutte le variabili stringa
3)Libera tutti gli spazi per le matrici
4)Resetta il puntatore di fine memoria
5)Resetta lo spazio di stack.

Non influenza quindi i contenuti del programma in memoria ma tutte le variabili, le matrici o tabelle, gli indirizzi di GO...SUB, i cicli di FOR...NEXT che sono riportati a zero, i valori attribuiti dalle funzioni utente (Vedi dopo). Anche la parte relativa ai files viene cancellata dalla memoria RAM del computer.
Ricordiamo che nel caso di file aperti, sia su nastro che su disco questi devono essere appropriatamente chiusi, magari con un comando diretto di CLOSE, perche' CLR non esegue correttamente la chiusura degli stessi.
Oltre al RUN anche i comandi di LOAD, DLOAD e naturalmente NEW, eseguono automaticamente un CLR.
Il comando CLR puo' essere eseguito sia in forma diretta che in programma. E' necessario ricordarsi, quando viene utilizzato all' interno di un programma Basic che questi continuera' la

sua esecuzione in base alle predette condizioni di ripulitura.

ESEMPIO:

```
100 A=12
110 PRINT A
120 CLR
130 PRINT A

RUN

     12

      0
```

In questo esempio si puo' vedere in forma semplice gli effetti dell' istruzione anche all' interno di un programma.

CMD

FORMATO: CMD numero di file logico[,lista]

FUNZIONE: Ridefinisce l' OUTPUT

AZIONE: L' uscita dei dati, cioe' l' OUTPUT, avviene di solito sullo schermo che appunto in questo caso sipuo' definire la periferica di DEFAULT per l' OUTPUT.
Con questo comando si puo' inviare l' OUTPUT su altra periferica come la stampante (di solito) o il disco. CMD deve essere seguito da un numero o da una variabile numerica che si riferisce al file. Il parametro lista puo' essere una stringa alfanumerica o una variabile.
CMD e' spesso utilizzato per eseguire i listati di programmi su stampante.

Ha la stessa lista di parametri del comando PRINT#

ESEMPIO: Creare un listato su stampante

OPEN 4,4

CMD4,"PROGRAMMA PIPPO"

LIST

PRINT#4 (annulla CMD)

CLOSE4

COLLECT

FORMATO: COLLECT [D(x)] [ON u(y)]

FUNZIONE: Libera tutti gli spazi che sono stati utilizzati con files scorrettamente chiusi su disco e toglie i relativi punti di riferimento dalla Directory del disco.

AZIONE: Un esempio di file impropriamente chiuso potrebbe essere quello di un file sul quale e' stato eseguito un comando OPEN ma MAI un comando CLOSE. In altre parole NON e' mai stato richiuso.
In questo caso il comando COLLECT DE-ALLOCA gli spazi occupati da questo file e ricrea la mappa di allocazione rimettendo a posto i FILES-LINK fra i rimanenti files.

NOTA

Viene usato anche il termine di SPLAT FILES per indicare quei files che appaiono sulla directory

del disco con un asterisco.

ESEMPI:

COLLECT D0 verifica il drive 0 sull' unita' 8

COLLECT verifica l' ultimo drive al quale si e' avuto accesso.

COLLISION

FORMATO: COLLISION tipo [,riferimento]

FUNZIONE: Definisce la manipolazione della funzione di interrupt per gli sprites

AZIONE: Il C 128 ha la possibilita', del resto gia' presente sul CBM 64,a nche trattabile in maniera piu' complessa, di rilevare quando qualcuno degli Sprites in movimento entra in contatto o in collisione con qualcosa di altro sia questo uno Sprite o un' altra immagine. Si possono infatti rilevare le collisioni degli Sprites e determinare poi quali Sprites sono entrati in collisione usando la funzione BUMP che vedremo.
Il comando COLLISION puo' individuare 3 tipi di eventi: collisione fra Sprites, collisione fra Sprites ed immagini sul video ed attivazione della penna ottica. Quando si verifica uno di questi eventi impiegando appunto questo comando il programma termina l' esecuzione della linea attuale e trasferisce il funzionamento, cioe' passa ad elaborare, il primo numero di linea della subroutine relativa alla gestione delle collisioni. In altre parole il funzionamento del programma SALTA, interrompendo quindi nel frattempo cio' che stava facendo, a quella serie

di istruzioni, che devono essere raggruppate in una subroutine, e che controllera' le collisioni. Dopo aver eseguito il RETURN che deve essere presente nella subroutine di gestione degli effetti di collisione il controllo del programma ritorna alla riga di programma successiva a quella che era stata interrotta.
I parametri che abbiamo indicato nel formato sono il TIPO di evento che in rapporto ai valori assegnati specifica quale tipo di evento dovrebbe dar luogo ad un interruzione. Vediamo i valori:

1 = Vengono individuate le collisioni fra Sprite e Sprite.

2 = Vengono individuate le collisioni fra Sprite ed una immagine qualsiasi dello schermo.

3 = Viene causato un interrupt all' attivazione della penna ottica (solo su 40 colonne).

Il parametro RIFERIMENTO e' il primo numero di linea della subroutine Basic a cui sara' trasferito il funzionamento del programma nel caso intervenga una collisione del tipo definito con il parametro precedente. Quando al parametro RIFERIMENTO viene assegnato un valore allora la funzione gestita da questo comando viene attivata, cioe' messa in ON per il tipo di collisione scelta.
Quando invece non viene assegnato nessun numero di linea allora il rilevamento di collisione per quel tipo di evento viene disabilitato, cioe' messo in OFF.
Si ha una collisione fra uno Sprite e l' altro quando parte di uno Sprite che non sia dello stesso colore del fondo viene ad occupare la stessa posizione di una parte di un' altro

Sprite anche questo di colore diverso dal fondo. Quando uno Sprite e' completamente fuori dello schermo e quindi nessuna delle sue parti e' visibile non puo' essere generata una collisione.
Avremo una collisione fra Sprite ed immagine quando una parte di uno Sprite che non sia dello stesso colore del fondo va a collidere con una parte di immagine qualsiasi, anche questa di diverso colore dal fondo, presente sullo schermo.
Gli Sprites che sono stati disattivati non possono causare collisioni.
Ricordiamo che piu' di un tipo di Interrupt per collisione possono essere attivati nello stesso momento, ma solo un tipo di collisione per volta puo' essere gestita. Vediamo con qualche esempio di chiarire ancora meglio questo comando.

ESEMPI

```
1000 COLLISION 1,500
```

Rileva una collisione fra Sprites ed il programma esegue un salto alla linea 500 che conterra' una subroutine di gestione della collisione.

```
100  COLLISION 1,1000
200  COLLISION 2,2000
....
....
1000 COLLISION:COLLISION 1
```

In questo esempio alla linea 100 avremo un trasferimento di esecuzione di programma, in pratica un GOSUB alla linea 1000 quando si verifica una collisione fra sprite e sprite.
Nella linea 200 avremo un salto a Subroutine di inizio a 2000 quando venga rilevata una

colisione fra Sprite ed immagineNella linea 1000 il rilevamento di ulteriori collisioni fra Sprites viene disattivato durante la gestione dell' attuale evento di collisione.
Come accennato si possono avere uno o piu' tipi di rilevamenti di collisioni attivi nello stesso tempo, ma solo una collisione per volta puo' essere gestita.
Ed e' per questo motivo che sarebbe utile disattivare successivi rilevamenti di collisione come primo passo nelle subroutine di gestione delle collisioni.
Questa operazione impedisce che intervengano altre interruzioni durante la gestione dell' interruzione corrente.
L' ultimo passo infine da eseguire nella subroutine di gestione della collisione sara' quello di riattivare il rilevatore di collisione.

NOTA

Per individuare con precisione quali sono gli sprites che sono entrati in collisione fra di loro o con immagini sullo schermo si puo' usare la funzione BUMP che vedremo in seguito.

COLOR

FORMATO: COLOR numero sorgente, numero colore

FUNZIONE: Definisce i colori per ogni area di schermo.

AZIONE:Sul C 128 e' possibile impostare indipendentemente le aree di sfondo, cornice e primo piano per uno dei 16 colori. Abbiamo inoltre a disposizione ben 7 aree di schermo

definibili con il primo parametro:

AREA	SORGENTE
0	Sfondo a 40 colonne
1	Cornice 40 colonne
2	Multicolor 1
3	Multicolor 2
4	Bordo a 40 colonne
5	Colore del carattere (40 o 80)
6	Colore di fondo a 80 colonne

Il secondo parametro identifica il colore e come vedremo dalle tabelle e' diverso a secondo se siamo in 40 o in 80 colonne.

CODICI COLORE NEL FORMATO A 40 COLONNE

1	Nero
2	Bianco
3	Rosso
4	Azzurro
5	Porpora
6	Verde
7	Blu
8	Giallo
9	Arancio
10	Marrone
11	Rosso chiaro
12	Grigio scuro
13	Grigio medio
14	Verde chiaro
15	Blu chiaro
16	Grigio chiaro

CODICI COLORE NEL FORMATO A 80 COLONNE

1	Nero
2	Bianco

3	Rosso scuro
4	Azzurro chiaro
5	Porpora chiaro
6	Verde scuro
7	Blu scuro
8	Giallo chiaro
9	Porpora scuro
10	Giallo scuro
11	Rosso chiaro
12	Azzurro scuro
13	Grigio medio
14	Verde chiaro
15	Blu chiaro
16	Grigio chiaro

ESEMPI

COLOR 4,2 fissa il colore del bordo a 40 colonne in Bianco

COLOR 5,9 fissa il colore del carattere visualizzato in Arancio se siamo a 40 colonne o in Porpora scuro se siamo ad 80 colonne.

CONCAT

FORMATO: CONCAT "file 2" [,D numero drive] TO "file 1" [,D numero drive] [(ON,)U periferica]

FUNZIONE: Concatena due files sequenziali.

AZIONE: Il vecchio file (file 2) e' cancellato e rimpiazzato con un nuovo file che e' la concatenazione dei due files. Tutte le volte che una variabile o un' espressione (che naturalmente deve essere risolta) e' utilizzata come nome del file, deve essere racchiusa fra

parentesi.

ESEMPIO: CONCAT"PIPPO" TO "PLUTO"

PLUTO diventa PIPPO+PLUTO

CONT

FORMATO: CONT

FUNZIONE: Consente di continuare l' esecuzione di un programma.

AZIONE: Questo comando fa ripartire l' esecuzione di un programma dopo che il tasto di RUN/STOP e' stato premuto o perche' e' stato incontrato nel programma un' istruzione di STOP o di END.
L' esecuzione del programma riparte dal punto in cui si era fermato.
Questo comando e' spesso utilizzato con il comando STOP per le prove o DEBUG. Infatti quando l' esecuzione di un programma viene fermata, i valori intermedi possono essere esaminati e cambiati utilizzando comandi in modo diretto.
Successivamente l' esecuzione del programma puo' essere ripresa con un CONT o con un comando GOTO in modo diretto, che fara' ripartire il programma da un dato numero di linea.
Non si puo' utilizzare il comando CONT ne' se il programma si e' fermato per un errore ne' se e' stato fatto un errore mentre si stava operando in modo diretto. In questo caso verra' visualizzato un messaggio di errore del tipo CANT' CONTINUE.

NOTA

In questo caso si ha un errore frequente e tipico quando si da il RETURN mentre il cursore e' posizionato sulla scritta READY. Infatti il Basic interpeta la sua stessa scritta come un comando READ Y su cui torneremo.

ESEMPIO:

```
100 A=0:B=1
110 A=A+8/B-8/(B+4)
130 PRINT A
140 B=B+8
150 GOTO 110
```

Questo programma calcola il valore di A e potra' essere una routine qualsiasi. Siccome e' attivato un ciclo di calcolo basato su un contatore se si desidera conoscere ad un momento qualsiasi il valore di A o di B sara' sufficiente premere lo STOP, digitare PRINT A oppure PRINT B esaminare il o i valori e dopo far continuare l' esecuzione del programma con CONT. In questo modo ne avremo arrestato l' esecuzione ma non alterato i contenuti fino a quel momento.

COPY

FORMATO:
COPY[D(x),]"(nome1)"TO[D(y),]"(nome2)"[ON U(z)]

Oppure: COPY[D(x)]TO[D(y)]

FUNZIONE: Esegue la copia di un file entro un' unita' a dischi.

AZIONE: Il comando COPY puo' funzionare sia fra 2 unita' a dischi che all' interno di un singolo drive.
Utilizzando questo comando senza nomi di files, la copia sara' effettuata sull' intero disco.
Variabili o espressioni che vengano utilizzate come nomi di files, devono essere racchiuse fra parentesi.

ESEMPIO:

COPY D0 TO D1

copia tutti i files dal disco nell' unita' 0 al disco nell' unita' 1.

COPY D1, "TESTO"TO"COPIA"

copia il file e ne cambia il nome.

DATA

FORMATO: DATA (lista di costanti)

FUNZIONE: Immagazzina costanti numeriche o stringa che saranno lette all' interno di un programma da comandi READ (vedi dopo).

AZIONE: I DATA non sono immediatamente eseguibili e possono essere immessi in qualsiasi punto del programma.
Un comando DATA puo' contenere piu' di una costante su di una stessa linea e possono esserci numerosi DATA in un programma.
Le costanti presenti nei DATA devono essere separate da virgole.
Il comando READ legge il contenuto dei DATA in maniera sequenziale in rapporto alla

progressione dei numeri di linea.
La lista delle costanti puo' contenere le costanti numeriche in qualsiasi FORMATO ò come interi, numeri in virgola mobile ecc., mentre non possono essere inserite espressioni da elaborare.
Per quanto riguarda le stringhe, anche queste intese sempre come costanti, non e' necessario che siano racchiuse fra virgolette a meno che non contengano virgole, due punti o spazi significativi.
Come abbiamo detto tutte la serie di DATA vengono trattate come lista continua. Le informazioni saranno quindi lette a partire dalla linea di programma con numero inferiore fino alla linea di programma contenente i DATA con numero maggiore e da sinistra verso destra.
L' istruzione READ legge i dati in rapporto al tipo richiesto (numeriche, stringa). Se cerchderemo di leggere un valore numerico ed incontriamo una stringa avremo un messaggio di errore.
Le informazioni contenute nei DATA possono essere rilette e quindi riutilizzate dall' inizio dando un comando di RESTORE in un conveniente punto del programma.

ESEMPIO

```
100 DATA 10,2,3.1415,50
110 READ A,B,C,D
120 PRINT A*2,B+7,C*2,D+3
```

In questo programma vengono letti i dati da destra a sinistra e stampati non come tali, ma come risultati di operazioni contenute nel comando alla linea 120.

DCLEAR

FORMATO: DCLEAR [D n.drive][(ON,)U periferica]

FUNZIONE: Chiude tutti i canali aperti sull' unita' a dischi

AZIONE: Questo comando esegue la chiusura di tutti i files ed esegue una funzione di CLEAR cioe' di pulizia o scarico su tutti i canali aperti di una unita' a disco.
I valori di DEFAULT sono rispettivamente per D 0 e per U 8. Puo' essere impiegato sia in forma diretta che in modo programma.

ESEMPI

DCLEAR Chiude tutti i files aperti sulla periferica 8 drive 0. E' equivalente a DCLEAR D0.

DCLEAR D0,U9 Operazione identica alla precedente ma sul drive 0 di una periferica alla quale e' stato assegnato il numero 9.

DCLOSE

FORMATO: DCLOSE[# numero di file logico] [(ON,)U periferica]

FUNZIONE: Esegue la chiusura di files disco

AZIONE: Questo comando puo' chiudere tutti i files aperti al momento su un' unita' a disco oppure solo il file logico dichiarato.
Se nessun numero di file e' specificato allora saranno chiusi tutti i files aperti in quel momento.

ESEMPI:

DCLOSE   Chiude tutti i files aperti in quel momento sull' unita' 8.

DCLOSE#5   Chiude il file associato con il file logico numero 5 sempre sull' unita' 8.

Vedi anche il comando DOPEN

DEF FN

FORMATO: DEF FN (nome) [(lista di parametri)] = (definizione della funzione)

FUNZIONE: Assegna un nome e definisce una funzione scritta dall' utente.

AZIONE: Il nome deve essere un nome di una variabile legale. Questo nome, preceduto da FN, diventa il nome della funzione.
La definizione di funzione e' un' espressione che permette l' operazione della funzione stessa.
I nomi di variabili che compaiono in questa espressione servono solamente per definire la funzione e non c' e' nessuna conseguenza sulle variabili di programma che hanno lo stesso nome.
.
Un nome di variabile utilizzato in una definizione di funzione puo' o non puo' apparire come parametro.
Se e' presente, allora il valore del parametro e' dato quando la funzione e' chiamata in esecuzione. In caso contrario viene utilizzato l' attuale valore della variabile.
Funzioni stringa definite dall' utente non sono

consentite.
Un comando DEF FN deve essere eseguito prima che la funzione da questi definita possa essere chiamata. Se la funzione viene richiamata dal programma prima che questa sia stata definita avremo una segnalazione di errore:

UNDEFINED FUNCTION.

Ricordiamo inoltre che DEF FN non e' utilizzabile in modo diretto ma solo all' interno di un programma.

ESEMPIO:

```
410 DEF FNAB(X)=X * 3/Y2
420 T=FNAB (I)
```

La linea 410 definisce la funzione FNAB, funzione che e' chiamata nella linea 420.

DELETE

FORMATO: DELETE [prima linea] [-ultima linea]

FUNZIONE: Cancella linee di un programma Basic in un intervallo dato.

AZIONE: Questo comando di utilita' alla programmazione serve per cancellare un certo numero di linee di programma in un programma Basic. Puo' essere utilizzato solo in modo diretto.
I parametri da assegnare sono simili a quelli per il comando LIST. Vediamone degli esempi.

ESEMPI

DELETE 30-100 Cancella tutte le linee di programma dalla 30 alla 100 comprese.

DELETE -100 Cancella tutte le linee di programma dall' inizio alla linea 100 compresa.

DELETE 700- Cancella tutte le linee di programma dalla 700 compresa fino alla fine del programma stesso.

DELETE 90 Cancella la linea di programma 90.

NOTA

Lo stesso risultato dell' ultimo esempio poteva essere ottenuto anche digitando il numero 90 e Return. Infatti digitando un numero di linea programma e facendolo seguire da un RETURN, cioe' un ritorno carrello a vuoto si ottiene come effetto la cancellazione della linea per annullamento.

DIM

FORMATO: DIM (lista di variabili)

FUNZIONE: Dimensiona i valori massimi per una matrice di variabili e ne predispone gli spazi di immagazzinamento in memoria.

AZIONE: Prima di poter usare una matrice di variabili, il programma deve eseguire un' istruzione di DIM cioe' di dimensionamento.Se una matrice di variabili e' usata senza un comando DIM, il numero massimo dei valori che puo' indirizzare e' da 0 a 10, cioe' di 11 elementi.
Se i valori usati sono piu' grandi del massimo specificato avremo un errore di "BAD SUSCRIPT".
Il comando DIM fissa tutti gli elementi di una data matrice ad un valore iniziale di ZERO.
Inoltre deve essere eseguita solo una volta per ciascun vettore perche' altrimenti avremo un errore di REDIM'D ARRAY. E' per questo motivo che la maggior parte dei programmatori eseguono il dimensionamento all' inizio della scrittura del programma.
Le matrici possono dimensionare qualsiasi tipo delle variabili, cioe' numeriche, intere o stringhe.
Per le variabili numeriche percio' metteremo una valida variabile numerica (A,B,AA,A1,ZX,U8, ecc.), mentre se vogliamo definire una matrice di dati alfanumerici aggiungeremo il segno dollaro ($) ed alle matrici di valori interi sara' aggiunto il segno percentuale % dopo il nome della variabile.
I numeri dopo il nome della variabile si chiamano indici ed il loro prodotto da luogo al totale degli elementi di una matrice.
Quindi il totale degli elementi di una matrice

puo' essere calcolato moltiplicando le dimensioni di ciascun indice per l' altro ricordando che si parte da 0.
Per esempio la matrice A(35) conterra' 36 elementi. La matrice A(6,9) conterra' 70 elementi (6+1)*(9+1). La matrice C1(9,20,5) conterra' 127 elementi (9+1)*(20+1)*(5+1).
Puo' essere quindi definita una matrice o tabella a singola entrata, doppia, tripla, quadrupla, ecc.
I limiti del numero di indici ed il valore massimo sono in funzione della memoria disponibile. Tuttavia il numero di indici e ci sembra ovvio che sia un limite teorico, non puo' superare i 255.

CONSUMO DELLA MEMORIA CON DIM

Riportiamo una tabella di consumo della memoria quando si utilizza un comando DIM ricordando che con ogni comando possono essere dimensionate piu' matrici.

5 Bytes per il nome della matrice
2 Bytes per ogni DIM
2 Bytes per ogni elemento di variabile INTERA (%).
5 Bytes per ogni elemento di variabile numerica.
3 Bytes per ogni elemento di variabile stringa
1 Bytes per ogni carattere in ogni elemento stringa.

NOTA

Osservate che ogni matrice a numeri INTERI occupa 2/5 dello spazio di una matrice con normali valori numerici.

ESEMPIO:

```
10 DIMA(20)
20 FORI=0TO20
30 READ A(I)
40 NEXT
50 DATA 1,2,3,....
```

Dimensionamento di matrice monodimensionale.

```
10 DIMR3(5,5)
20 DIMD$(2,2,2)
```

Nella linea 10 e' dimensionata una matrice bidimensionale.
Nella linea 20 invece la matrice e' a tre dimensioni.

DIRECTORY

FORMATO: DIRECTORY [D n.drive] [,(ON,)U periferica] [,widcard]

FUNZIONE: Visualizza la Directory del disco sullo schermo.

AZIONE Con questo comando si puo' visualizzare la Directory del dischetto o dei dischetti se abbiamo collegato 2 unita'. Nel modo 128 abbiamo la possibilita' di eseguire questa funzione semplicemente premendo il tasto funzione F3 e verra' visualizzat la Directory per la periferica 8 drive0.
Per fermare temporaneamente la visualizzazione utilizzare il tasto CONTROL S o NO SCROLL. Per ripartire un tasto qualsiasi. Il tasto con il simbolo COMMODORE serve per diminuire la velocita' di scorrimento. Questo comando non

cancella il programma in memoria, mentre impiegando l' altro sistema (LOAD"$0",8) si cancella il programma eventualmente in memoria.
Il parametro WILDCARD indica il o i nomi di files, magari con abbreviazione che si vogliono listare.

ESEMPIO:

DIRECTORY Lista tutti i files su disco

DIRECTORY"CA*" Lista tutti i files che iniziano con le parole CA, per esempio i capitoli di questo volume.

DIRECTORY (C$) Visualizza tutti i files contenuti nella variabile C$.

DLOAD

FORMATO: DLOAD"nome file"[,D n.drive][,ON U periferica]

FUNZIONE: Carica un file programma da disco

AZIONE: Questo comando e' normalmente utilizzato per caricare un programma Basic dall' unita' a dischi 8, drive 0.
Il nome del file puo' essere una variabile stringa o una stringa fra virgolette.
Un modo di utilizzare questo comando e' in forma diretta ed abbreviata cioe' premendo il tasto SHIFT contemporaneamente al RUN/STOP. Metodo con il quale si carica il primo programma del drive 0.
DLOAD puo' essere utilizzato anche in modo programma sempre per richiamare un programma da disco.

ESEMPIO:

DLOAD"PIPPO"

DLOAD"PIPPO",D1 ON U9

Mentre nel primo esempio viene cercato il file PIPPO in un drive qualsiasi, nel secondo esempio la ricerca avviene SOLO nel drive 1 dell' unita' a dischi 9.
Fra l' altro si ricorda che il numero della periferica puo' essere cambiata sia via HARDWARE che SOFTWARE.

100 DLOAD(B$)

Carica cioe' un programma da disco il cui nome e' stato precedentemente inserito nella variabile B($). Se la variabile B($) non riporta nessun valore, cioe' e' vuota, verra' visualizzato un messaggio di errore.

NOTAF

Come abbiamo detto questo comando puo' essere usato in modo programma. Quando viene usato in questo modo, cioe' dall' interno di un programma per richiamarne un' altro, allora siamo in presenza di una tecnica chiamata CHAINING.

DO/LOOP/WHILE/UNTIL/EXIT

FORMATO: DO[UNTILcondizione/WHILE condizione] istruzioni [EXIT] LOOP [UNTIL condizione/WHILE condizione]

FUNZIONE: Definisce e controlla un ciclo di programma.

AZIONE: Esegue le istruzioni comprese fra DO e LOOP. Se ne UNTIL ne' WHILE modificano l' istruzione DO o LOOP allora l' esecuzione delle istruzioni contenute nel ciclo continua senza fine.
Se si incontra una istruzione di EXIT durante l' esecuzione di un ciclo DO LOOP, allora l' esecuzione del programma passa all' istruzioneche segue LOOP.
I cicli DO si possono nidificare seguendo le regole relative ai cicli FOR...NEXT.
Se si utilizza il parametro UNTIL, il programma continua ad eseguire cicli fino a che non si abbia una condizione di VERO. Il parametro WHILE e' l'opposto di UNTIL per cui il programma continua a girare fino a quando permane la condizione di verita'.

ESEMPI

```
10  PRINT"HO PENSATO UN NUMERO INDOVINA"
20  X=INT(RND(1)*10)
30  IFX=0THEN20
40  DO WHILE A<>X
50  INPUT A
60  IF A=9999THEN EXIT
70  IF A<>XTHEN PRINT"ERRORE":N=N+1
80  LOOP
90  IFA=9999THEN120
100 PRINT"HAI INDOVINATO DOPO"N" TENTATIVI"
110 END
120 PRINT "HAI ABBANDONATO"
```

Si tratta di un programma simile a aquello visto in precedenza. Vi invitiamo a farlo girare e sopratutto ad apportarvi tutte quelle variazioni per comprendere a fondo questo sistema.

DOPEN

FORMATO: DOPEN# n. di file logico,"nome del file"[,(S/P)][,L lunghezza record] [,D n. drive] [ON,U periferica] [,W]

FUNZIONE: Apre un file su disco per un' operazione di lettura o scrittura.

AZIONE: Questo comando apre su disco un file sequenziale, relativo o ad accesso casuale (RANDOM) per un' operazione che potra' essere di lettura o scrittura.
I parametri utilizzati nel comando sono i seguenti:

S = File di tipo sequenziale
P = File di tipo programma
L = Lunghezza del record (solo per Relatives)
W = Per operazioni di scrittura.

Relativamente a quest' ultimo parametro ricordiamo che se non viene dichiarato si potra' dar luogo solo ad operazione di lettura.
Il numero di file logico utilizzato deve essere compreso in un' intervallo fra 1 e 255.
Un numero di file logico maggiore di 128 forza un ritorno carrello ed un line feed dopo ogni comando PRINT#.
Numeri di file logico inferiori a 128 inviano solo un ritorno carrello.
Il ritorno carrello puo' essere annullato con l' uso di un punto e virgola.
Se e' assente il parametro relativo all' unita a dischi questa sara' assunta pari ad 8, mentre se manca il parametro relativo al drive, questi sara' messo pari a 0 (zero).
Come abbiamo gia' detto per quanto riguarda le

operazioni su files sequenziali dovra' essere specificato il parametro W per operazioni di scrittura (WRITE), perche' in caso contrario il file sara' aperto solo per operazioni di lettura (READ).
Quando una variabile o una espressione da calcolare e' utilizzata come nome del file dobbiamo usare le parentesi.

E' possibile rimpiazzare un file esistente con il comando DOPEN utilizzando la a commerciale:
DOPEN#2,"@FILE 1",D1

ESEMPIO:

DOPEN#5,(A$)

Dove A$= "nome del file"

DRAW

FORMATO: DRAW [colore sorgente], X1,Y1, [TO X2,Y2]...

FUNZIONE: Disegna punti linee e figure in una data posizione di schermo.

AZIONE: Questo comando che puo' essere utilizzato sia in forma diretta che in modo programma e' uno dei piu' utili per la grafica perche' consente di disegnare sia punti che linee e quindi forme di qualsiasi tipo. I parametri sono:

COLORE SORGENTE i cui valori possono essere i seguenti:

0 = Sfondo
1 = Primo piano
2 = Multicolor 1
3 = Multicolor 2

X1 e Y1 sono le coordinate di partenza

X2 e Y2 le coordinate di fine

Se il colore sorgente viene omesso allora per disegnare verra' usato il colore di primo piano.
Il disegno inizia dalla posizione indicata dalle coordinate X1 e Y1.
Se questo parametro viene omesso allora il disegno inizia dalla attuale posizione del PC.
Viene cosi' disegnata una linea fino all' altro punto di coordinate X2 e Y2. Se questi ultimi parametri sono omessi verra' disegnato un' unico punto.
Il disegno di una linea sposta il PC sull' ultimo punto tracciato.
Come per gli altri comandi grafici visti, si possono usare, per le coordinate sia valori assoluti che relativi.
Inoltre per i parametri omessi si deve impiegare una virgola.

ESEMPI

DRAW 1,100,50 Disegna un punto nel colore del primo piano

DRAW 0,100,50 Cancella il punto disegnato in precedenza.

DRAW TO 200,100 Disegna una retta che va dall' attuale posizione del PC fino ai valori specificati.

DSAVE

FORMATO: DSAVE"nome del file" [,D n. drive] [(ON,)U n. periferica]

FUNZIONE: Salva un programma Basic su disco.

AZIONE: Con questo comando si puo' trasferire un programma in memoria su dischetto.
Il nome del file specificato come parametro non deve essere superiore a 16 caratteri, mentre non dichiarando ne' l' unita' ne' il drive, queste verranno assunte rispettivamente come 8 e 0.
Quando come nome del file siano utilizzate variabili o espressioni da elaborare, queste devono essere racchiuse fra parentesi.
E' possibile rimpiazzare un file esistente utilizzando la a commerciale (@) prima del nome del file da registrare.

ESEMPIO:

```
DSAVE"PROGRAMMA"

DSAVE"@PROGRAMMA",D1
```

Nel primo caso viene salvato un programma il cui nome comunque non esisteva nella directory di quel dato disco.
Nel secondo caso il programma viene RIMPIAZZATO.

DVERIFY

FORMATO: DVERIFY"nome del file" [,D n. drive] [(ON,)U n. periferica]

FUNZIONE: Confronta il programma in memoria con quello sul disco.

AZIONE: Con questo comando il C 128 controlla che il programma in memoria in quel momento sia uguale a quello su disco. Naturalmente anche in questo caso i valori DEFAULT sono 0 per numero di drive e 8 per il numero di periferica.

ESEMPI:

DVERIFY "PIPPO"

Controlla che il programma su disco di nome PIPPO sia uguale al programma attualmente in memoria.

DVERIFY "PIPPO",D0,U9

Controlla come in precedenza ma sulla periferica 9 del drive 0.

NOTA

Se un' area grafica e' allocata o riallocata dopo un SAVE avremo una segnalazione di errore. Infatti malgrado l' operazione in se sia corretta bisogna ricordare che durante le operazioni grafiche i puntatori della memoria vengono spostati e pertanto l' operazione di verifica che viene fatta con confronto Byte per Byte non puo' dare che un errore.
Per verificare i Files Binari vedi il comando VERIFY.

END

FORMATO: END

FUNZIONE: Termina l' esecuzione di un programma e riporta il computer a livello comandi.

AZIONE: Il comando END puo' essere immesso in qualsiasi punto del programma per terminarne l' esecuzione.
A differenza del comando STOP, l' END non causa la visualizzazione di un messaggio di BREAK, ma solo del READY.
Comunque un END al termine di un programma e' opzionale e mai usato tranne che in sede di prove o di controllo di sviluppo.
All' interno di un programma ci possono essere diverse istruzioni di END. Il comando CONT puo' essere usato per far ripartire il programma ammesso che dopo l' END incontrato ci siano altre linee Basic da elaborare.

ESEMPIO:

```
520 IF K=1000 THEN END
```

In questo caso quando la variabile numerica K diventa pari al valore 1000 il programma si ferma.

ENVELOPE

FORMATO: ENVELOPE n,[,atk] [,dec] [,sus] [,rel] [,wf] [,pw]

FUNZIONE: Determina le forme d' onda e gli INVILUPPI dei suoni.

AZIONE: Per utilizzare piu' facilmente l' integrato SID (SOUND INTERFACE DEVICE) ecco il primo dei comandi musicali. Vediamone di seguito i parametri con i valori che possono essere loro

assegnati.

N Numero di inviluppo (0-9)
ATK Livello dell' attacco (0-15)
DEC Livello di decadimento (0-15)
SUS Livello di sostegno (0-15)
REL Livello di rilasco (0-15)
WF Forma d' onda del suono che potra' essere:

0 = Triangolo
1 = Dente di sega
2 = Onda quadra o impulso
3 = Rumore di fondo
4 = Modulazione ad anello

PW Ampiezza d'onda (0-4095) che ha significato solo quando la forma d' onda scelta e' 2.

Vediamo un po' di chiarire.
ATTACCO e' il tempo durante il quale il suono passa dal volume minimo al volume massimo.
DECADIMENTO e' il tempo che il suono impiega per passare dal volume massimo al livello di sostegno.
SOSTEGNO e' il livello di volume a cui la nota sara' tenuta per la maggior parte della sua durata.
RILASCIO e' il tempo che il suono impiega per passare dal livello sostegno a zero fino cioe' ad annullarsi completamente.
Per ottenere i suoni relativi a strumerenti musicali conosciuti sono gia' stati predefiniti 10 diversi INVILUPPI per i quali oltre naturalmente al comando sara' sufficiente dare il numero.
La seguente tabella li riporta con i valori degli altri parametri che sono gia' stati

assegnati e che potranno servire da guida per eventuali cambiamenti pur nell' ambito dello strumento stesso.

N	ATK	DEC	SUS	REL	WF	PW	STRUMENTO
0	0	9	0	0	2	1536	piano
1	12	0	12	0	1		fisarmonica
2	0	0	15	0	0		calliope
3	0	5	5	0	3		tamburo
4	9	4	4	0	0		flauto
5	0	9	2	1	1		chitarra
6	0	9	0	0	2	512	clavicembalo
7	0	9	9	0	2	2048	organo
8	8	9	4	1	2	512	tromba
9	0	9	0	0	0		xilofono

FAST

FORMATO: FAST

FUNZIONE: Mette il sistema in condizione di funzionare a 2 MHz

AZIONE: Nel C 128 il microprocessore 8502 ha la possibilita' di funzionare sia a 1 MHz che a 2 MHz cioe' a velocita' operativa interna doppia. Benche' cio' non abbia alcun effetto sulle operazioni di I/O la velocita' di esecuzione delle operazioni presenti in un programma Basic aumenta in modo considerevole.
Possono essere impiegati anche i comandi grafici, tuttavia questi saranno invisibili sul Monitor fino a quando non venga ripristinata la normale velocita' di esecuzione con il comando SLOW.

FETCH

FORMATO: FETCH #bytes,intsa,expb,expsa

FUNZIONE: Legge i dati da un' espansione di memoria esterna.

AZIONE: Vediamone i parametri ed i valori che a questi possono essere assegnati.
BYTES E' il numero di Bytes che devono essere letti dall' espansione di memoria da 1 a 65535

INTSA E' l' indirizzo di inizio della memoria RAM INTERNA (0-65535)

EXPSA E' l' indirizzo di inizio dell' espansione di memoria (0-65535)

EXPB E' il numero del banco di memoria che si assegna all' espansione esterna.

Il funzionamento di questo comando e degli altri che si riferiscono a questa particolare gestione della memoria esterna saranno visti nella sezione dedicata alla gestione della memoria.

FILTER

FORMATO: FILTER [freq] [,lp] [,bp] [,hp] [,res]

FUNZIONE: Definisce i parametri di filtro nel suono.

AZIONE: Questo comando viene usato per variare dinamicamente le qualita' di tono del suono prodotto. Con FILTER si aziona un filtro d' onda presente sul SID per sopprimere le gamme di

frequenza scelte.
Si potra' anche specificare un' effetto di risonanza che enfatizzi le note con frequenze vicine a quelle di taglio del filtro. Vediamo i parametri ed i valori assegnabili:

FREQ Frequenza di taglio del filtro (0-2048)
LP Attivazione del filtro passa-basso 0=OFF 1=ON
BP Attivazione del filtro passa-banda 0=OFF 1=ON
HP Attivazione del filtro passa-alto 0=OFF 1=ON
RES Risonanza (0-15)

Il parametro FREQ e' la frequenza di taglio per il filtro nel SID ed il suo valore puo' variare da 0 a 2048. Per determinare l' effettiva frequenza di taglio in Hz sara' necessario moltiplicare questo valore per una costante fissa 5.8 e aggiungere 30.
Gli altri tre parametri vengono usati insieme per determinare quali parti dello spettro audio debbono passare inalterate, cioe' senza alcuna modificazione dall' uscita del SID e quali parti, se sono in ON, devono essere tagliate dal filtro. Infatti ognuno di questi parametri puo' avere un valore 0 o 1 e cioe', rispettivamente soppressione o passaggio. Si potranno impostare uno o piu' parametri con differenti valori a secondo di cio' che si vuole ottenere.
Il parametro RES, cioe' la risonanza che puo' variare da 0 a 15 e determina la risonanza dell' uscita sonora. In altre parole quando venga enfatizzato o meno l' effetto massimo dei suoni in prossimita' della frequenza di taglio.

ESEMPI:

100 FILTER 1048,1   Imposta il valore di taglio del filtro ed il filtro passa-basso

100 FILTER 2048,,,8 Imposta il valore di taglio ed il controllo di risonanza

500 FILTER 2048,1,0,1,8 Imposta le modalita' di taglio ed seleziona i filtri passa-basso e passa-alto per avere un risultato di NOTCH REJECT. Fissa la risonanza a livello 8.

NOTA

Ricordiamo che per ottenere un effetto udibile con l' uso dei filtri almeno un tipo di questi deve essere selezionato ed almeno una voce deve operare su quel filtro.

FOR NEXT

FORMATO: FOR (variabile)=(x)TO(y)[step(z)]
.
.
.
NEXT[(variabile)][,(variabile)...]

dove x,y,z sono epressioni numeriche.

FUNZIONE: Permette che una serie di istruzioni possano essere messe in un ciclo, cioe' ripetute un dato numero di volte.

AZIONE: La variabile e' usata come contatore. La prima espressione numerica (x) e' il valore iniziale del contatore, mentre la seconda epressione numerica (y) sara' il valore finale del contatore.
Le linee di programma che seguono il comando FOR vengono eseguite fino a quando non sia incontrato il NEXT. Il contatore viene incrementato di 1 o dal valore specificato nel parametro STEP.
Viene eseguito quindi un controllo per vedere se il valore presente nel contatore sia a questo punto maggiore di quello del valore finale (y).
Se non e' maggiore il Basic ritorna indietro all' istruzione che segue immediatamente il comando FOR e quelle sezione di progrramma e' ripetuta.
Se invece e' maggiore, l' esecuzione del programma continua con il comando seguente il NEXT.
In breve sintesi questo e' un ciclo di FOR...NEXT.
Se il parametro STEP (cioe' il passo) e' negativo, il valore finale del contatore e' fissato per essere minore del valore iniziale.
In questo caso il contatore invece di essere

incrementato e' DECREMENTATO tutte le volte che il ciclo viene eseguito ed il confronto verra' fatto naturalmente non sul maggiore, ma sul minore del valore finale che deve assumere il contatore stesso.

CICLI NIDIFICATI

I cicli di FOR...NEXT possono essere nidificati cioe' inseriti uno dentro l' altro. In altre parole un ciclo di FOR...NEXT puo' essere immesso all' interno di un' altro ciclo di FOR...NEXT.
Quando i cicli sono nidificati, ogni ciclo deve avere un nome di variabile unico come suo contatore.
Puo' invece essere utilizzato un solo comando di NEXT per tutti i cicli, a condizione che sia seguito da tutti i nomi delle variabili e che questi nomi siano in ordine di esecuzione e separate da virgole.
Questo come regola generale del Basic. Tuttavia nel Basic Commodore le variabili del comando NEXT possono essere omesse, caso in cui il comando NEXT si riferira' al piu' vicino FOR.
Nel caso venga trovato nell' ambito del programma un NEXT PRIMA del suo corrispondente FOR, avremo una segnalazione di errore del tipo:

NEXT WITHOUT FOR

ed il programma si interrompera'.
Inoltre, a causa della limitazione dell' area di STACK che serve ad immagazzinare i punti di ritorno, un ciclo nidificato non puo' contenere piu' di 9 cicli FOR...NEXT.

Vediamo ora qualche esempio commentato.

ESEMPIO 1: Ciclo nidificato.

```
10 FOR I=1TO3
20 FOR J=1TO3
30 PRINT I;J
40 NEXTJ,I
```

RUN

```
 1  1
 1  2
 1  3
 2  1
 2  2
 2  3
 3  1
 3  2
 3  3
```

ESEMPIO 2:

```
10 K=10
20 FORI=1TOK STEP 2
30 PRINT I;
40 K=K+10
50 PRINT K
60 NEXT
```

RUN

```
 1  20
 3  30
 5  40
 7  50
 9  60
```

ESEMPIO 3: Secondo valore minore del primo.

```
10 J=0
20 FORI=1TOJ
30 PRINT I
40 NEXT I

RUN

     1
```

NOTA

In questo esempio il ciclo sara' eseguito una sola volta perche' il valore iniziale del ciclo stesso e' superiore al valore finale, ma non e' controllato fino a quando non sia stato eseguito il NEXT.

ESEMPIO 4: Variabili usate preventivamente.

```
10 I=5
20 FOR I=1 TOI+5
30 PRINTI;
40 NEXT

RUN

1  2  3  4  5  6

Ready.
```

In questo esempio il ciclo e' eseguito 6 volte. Il valore iniziale della variabile del ciclo era stata fissata prima del valore finale.

NOTA

Ricordiamo che non si puo' utilizzare una variabile INTEGER (I%) come contatore di variabile.

Esempio:

```
10 FOR I% =1TO10
20 PRINT I%
30 NEXT%

RUN

?SYNTAX ERROR IN 10
```

GET

FORMATO: GET [#(file logico),] (variabile)

FUNZIONE: Legge un carattere da un file entro una variabile.

AZIONE: GET a solo, cioe' senza un numero di file logico esegue la scansione del buffer di tastiera e riporta un valore numerico o stringa se il buffer contiene un segno di tasto premuto. Un riporto nullo per variabili numeriche e' 0, mentre per variabili stringa e' LEN(a$)=0.
GET# legge un carattere dal file logico specificato. Se la periferica dichiarata nel comando OPEN e' 0 allora il GET# si comportera' come un GET normale.
Se il file logico dichiarato e' 1 o 2 allora il comando GET# eseguira' un ritorno carrello o una condizione di EOF (End of File), che potranno essere esaminate controllando ST.

ESEMPIOs:

```
10 PRINT" PREMERE UN TASTO"
20 GET A$:IF A$=""THEN 20
```

GETKEY

FORMATO: GETKEY lista di variabili

FUNZIONE: Riceve dati da tastiera, un carattere per volta.

AZIONE: Questo comando e' molto simile al comando GET visto in precedenza, ma differenza di quello GETKEY e' specifico per l' uso della tastiera. Infatti attende che sia premuto un carattere da tastiera per continuare la sua esecuzione.
Gli esempi chiariranno meglio.

ESEMPI

```
100 GETKEY B$
```

Quando il programma arriva a questa linea si ferma ed attende che sia premuto un tasto qualsiasi per proseguire la sua esecuzione.

```
100 GETKEY Z$,C$,F$
```

Simile al precedente ma che dimostra come si possa usare il comando in forma multipla. In questo caso infatti il programma attende che siano battuti 3 caratteri prima di proseguire.

NOTA

Ricordiamo che questo comando puo' essere

eseguito solo in modo programma.

GO 64

FORMATO: GO64

FUNZIONE: Passa al modo CBM64

AZIONE: Questo computer incorpora ANCHE un CBM 64 con tutte le relative funzioni e questo comando consente quindi di passare via software da un modo di funzionamento all' altro.
In risposta a questo comando dato in modo diretto viene visaulizzato un:

Are You Sure?

Al quale si dovra' rispondere con Y per Yes se realmente si vuol passare all' altro modo di operare del sistema. Si tratta di un controlo di sicurezza necessario perche' durante questo passaggio TUTTI i dati presenti nella memoria del computer vengono irrimediabilmente persi.
Questo comando puo' essere utilizzato anche in modo programma , ma in questo caso non verra' visualizzata la precedente domanda.

GOSUB...RETURN

FORMATO: GOSUB (numero di linea)
.
.
.
RETURN

FUNZIONE: Esegue un salto ed un riturno da

subroutine.

AZIONE: Per NUMERO DI LINEA si intende la prima linea della subroutine.
Il comando RETURN in una subroutine ordina al Basic di tornare indietro e precisamente al comando seguente il piu' recente GOSUB.
Una subroutine puo' contenere piu' di un comando RETURN, dovrebbe logicamente consentire dei riposizionamenti a differenti punti della subroutine.
Una subroutine puo' apparire in un punto qualsiasi del programma, ma e' distinguibile dal programma principale.
Per prevenire accessi inavvertiti alla subroutine, questa puo' essere preceduta da comandi STOP, END o GOTO che dirigano il controllo di programma attorno alla subroutine.
Un incremento nella velocita' di esecuzione puo' essere osservata assegnando numeri bassi alle subroutines e che precedano quindi il corpo del programma.
Allo stesso modo che abbiamo visto per i cicli nidificati, possibili anche in questo caso cioe' a livello di subroutines, ogni GOSUB deve essere seguito da un RETURN e, sempre a causa della limitata capienza dell' area di STACK, il massimo numero di GOSUB utilizzabili e' di 23.

ESEMPIO:

```
10 GOSUB 40
20 PRINT"RITORNO DA SUBROUTINE"
30 END
40 PRINT"SUBROUTINE";
50 PRINT"IN";
60 PRINT"ESECUZIONE";
70 RETURN

RUN
```

```
SUBROUTINE IN ESECUZIONE
RITORNO DA SUBROUTINE

Ready.
```

GOTO

FORMATO: GOTO (numero di linea)

FUNZIONE: Salta in modo incondizionato fuori dalla normale sequenza del programma ad un dato numero di linea.

AZIONE: Se il numero di linea al quale rimanda l' istruzione contiene un comando eseguibile, allora sia quel comando che i seguenti vengono eseguiti.
Se invece viene incontrato un comando non eseguibile, l' esecuzione del programma procede oltre fino al primo comando eseguibile dopo il numero di linea specificato nel parametro.

ESEMPIO:

```
10 READ R
20 PRINT"R=";R,
30 A=3.14*R*R
40 PRINT"AREA=";A
50 GOTO 10
60 DATA 5,7,12

RUN

R=5        AREA=78.5
R=7        AREA=153.86
R=12       AREA=425.16
```

```
?OUT OF DATA ERROR IN 10

READY
```

NOTA

Ricordiamo che questo comando puo' essere dato anche nella forma staccata GO TO.

GRAPHIC

FORMATO 1: GRAPHIC modo [,clear][,s]

FORMATO 2: GRAPHIC CLR

FUNZIONE: seleziona uno dei modi grafici
AZIONE: Questo comando consente di selezionare uno dei 6 modi grafici del C 128 in funzione del valore dato al parametro MODO che puo' quindi assumere uno dei seguenti valori:

MODO	DESCRIZIONE
0	Testo normale a 40 colonne
1	Alta risoluzione standard
2	Alta risoluzione SPLIT-SCREEN
3	Multi-color
4	Multi-color SPLIT-SCREEN
5	Testo a 80 colonne

Il parametro opzionale CLEAR specifica appunto una pulizia CLR di schermo dopo che il programma abbia girato. Questa funzione sara' eseguita se CLEAR e' = 1 oppure non sara' eseguita se e' invece = 0.
Il parametro S serve ad indicare il numero di partenza dello SPLIT-SCREEN ed e' significativo solo quando abbiamo selezionato in precedenza i

modi grafici 2 o 4. Il valore di DEFAULT per questo parametro e' la linea 19, cioe' rimangono visibili in modo testo, ricordiamo che lo SPLIT-SCREEN e' infatti un modo misto di testo e grafica, le ultime 5 linee.
Particolarita' di questo comando.e' che quando esso e' eseguito nei modi da 1 a 4 viene riservata un' area di 9 K per lavorarci.
L' inizio dell' area testo del Basic viene spostata oltre quest' area e quindi ogni programma Basic viene automaticamente rilocato nella nuova posizione di memoria.
Quest' area resta riservata anche se si ritorna in modo testo e come abbiamo gia' visto, se al parametro CLEAR si da il valore 1 lo schermo viene cancellato.
La seconda forma del comando che abbiamo visto e cioe' GRAPHIC CLEAR rende nuovamente disponibile per il testo Basic e per le variabili l' area di 9 K che avevamo riservato in precedenza e riposiziona il programma Basic eventualmente presente in memoria e comunque i relativi puntatori nella posizione standard.
Un altro modo per cancellare lo schermo senza usare il comando GRAPHIC e' quello che vedremo poi di usare il comando SCNCLR che viene usato da solo e senza parametri e che ha lo stesso effetto di assegnare un valore 1 al parametro CLEAR di GRAPHIC.
Ricordiamo inoltre che GRAPHIC puo' essere usato sia direttamenteche in modo programma.

ESEMPI

GRAPHIC 0 Seleziona il modo 40 colonne

GRAPHIC 5 Seleziona il modo 80 colonne

GRAPHIC3,1 Seleziona il modo MULTI-COLOR ed esegui la pulizia dell' area grafica.

GRAPHIC2,1,15 Seleziona il modo alta risoluzione con SPLIT-SCREEN, esegui la pulizia dell' area grafica e fai iniziare l' area di SPLIT-SCREEN dalla riga 15.

HEADER

FORMATO: HEADER"(nome disco)",D(x)[,I(zz)][ON U(y)]

FUNZIONE: formatta un nuovo disco o esegue la pulizia su un disco vecchio.

AZIONE: Quando il parametro I(zz), cioe' il numero di Identificatore Disco, e' dichiarato esso non solo viene assegnato al dischetto stesso, ma viene anche eseguita PER INTERO l' operazione di formattazione.
In caso contrario viene eseguita solo la pulizia della Directory ed un nuovo nome e' assegnato al dischetto stesso.
Questo comando richiede una certa cautela nell' utilizzo.
Infatti il sistema chiede:

ARE YOU SURE?

prima di eseguire l' operazione prprio per il carattere distruttivo della stessa.
A questa domanda va risposto digitando Y per YES se si desidera confermare oppure premere qualsiasi altro tasto per annullare l' operazione.
Inoltre nell' esecuzione di HEADER e' possibile rilevare degli errori dovuti principalmente o a dischetti completamente rovinati o con malformazioni nell' area di scrittura della

Directory o a dischetti protetti o altro.
Il comando HEADER eseguira' una lettura del canale comando dell' unita'a a dischi e se rileva un errore, riportera' un messaggio: "?BAD DISK".
A questo punto e' bene controllare che il dischetto non sia protetto, resettare il computer e rieseguire l' operazione dall' inizio. Nel caso si ripeta l' errore sara' bene cambiare dischetto.
Quando una variabile o una espressione e' utilizzata come nome del disco deve essere racchiusa fra parentesi.
Ricordiamo che l' Identificatore Disco non puo' essere dichiarato con una variabile.

ESEMPIO:

HEADER"DISCO CBM",D0,I01

NOTE

Questo comando equivale a OPEN 1,8,15,"N0:Nome del disco, identificatore" del Basic del CBM64.

HELP

FORMATO: HELP

FUNZIONE: evidenzia un errore

AZIONE: Questo comando e' utilizzato in modo diretto quando c'e' un errore in un programma e questi e' stato segnalato.
Quando siamo in modo 40 colonne allora la parte della linea di programma che contiene l' errore e' visualizzata in reverse. Quando invece siamo in modo 80 colonne la parte del programma

contenente l' errore e' sottolineata.

IF...THEN e IF...GOTO

FORMATO: IF (espressione) THEN (comando)(numero di linea).

FORMATO: IF (espressione) GOTO (numero di linea).

FUNZIONE: Serve a prendere decisioni riguardanti il flusso del programma, decisioni che si basano sul risultato di un' espressione.

AZIONE: Se il risultato dell' espressione e' diverso da 0, allora sono eseguiti il THEN o il GOSUB.
THEN puo' essere seguito da un numero di linea che indichi il punto dove deve saltare o da uno o piu' comandi che devono essere eseguiti. GOTO invece deve essere sempre seguito da un numero di linea.
Se il risultato dell' espressione che segue l' IF e' zero allora le disposizioni di THEN e GOTO sono ignorate e l' esecuzione del programma continua con il successivo primo comando eseguibile della linea seguente.
I comandi IF...THEN come abbiamo visto gia' per il FOR...NEXT, possono essere nidificati e l' unica limitazione e' data dalla lunghezza della linea.

```
IF A=B THEN IF B=C THEN PRINT"A=C"
```

Se un comando IF...THEN e' seguito da un numero di linea in modo diretto, verra' riportato un

errore di:

UNDEFINED STATEMENT

a meno che il programma Basic presente non contenga quel numero di linea.

NOTA

Quando si usa IF per controllare l' eguaglianza di un valore che sia il risultato di un' operazione in virgola mobile, ricordiamoci che la rappresentazione interna del valore puo' non essere esatta.
Per questo il controllo dovrebbe essere effettuato nell' intervallo nel quale l' accuratezza del valore deve variare.
Per esempio, per controllare il valore calcolato della variabile A entro il valore 1.0 si deve usare:

```
IF ABS(A-1.0)<=1.0E -6THEN....
```

ESEMPIO:

```
200 IF I THEN GET I
```

Questo comando esegue la scansione di tastiera per il numero I se I e' diverso da zero.

ESEMPIO:

```
100 IF(I>10)AND(I<20)THEN DB=1979-1:GOTO300
110 PRINT"FOURI DELL' INTERVALLO"
.
.
.
```

In questo esempio il controllo determina se I e' maggiore di 10 e minore di 20. Se I e' in questo intervallo, DB e' calcolato e l' esecuzione del programma salta alla linea 300.
Se invece I non e' in questo intervallo, l' esecuzione continua con la linea 110.

INPUT

FORMATO: INPUT ["stringa";](lista delle variabili)

FUNZIONE: Consente una immissione di dati da tastiera durante l' esecuzione di un programma.

AZIONE: Il comando INPUT e' illegale, cioe' non puo' essere dato, in modo diretto.
Quando viene incontrato un comando INPUT durante l' esecuzione di un programma, questi si ferma, un punto interrogativo viene visualizzato per indicare che il programma sta attendendo dei dati e vicini al punto interrogativo e' presente il cursore lampeggiante.
Se dopo il comando INPUT e' inserita un stringa, fra virgolette, questa sara' stampata prima del punto interrogativo ed i dati saranno successivamente inseriti.
I dati che sono inseriti sono assegnati alla variabile o variabili presentati nella lista.
Il numero di dati forniti deve essere lo stesso del numero di dati nella lista delle variabili.

Piu' dati nella stessa linea di INPUT devono essere separati da virgole.
INPUT e' limitato dalla lunghezza della linea logica di schermo che e' attuale mente di 78 caratteri.
Nel caso siano inseriti piu' caratteri saranno accettati solo quelli che rientrano nella linea logica di schermo e gli altri saranno ignorati.
I nomi delle variabili nella lista possono essere numerici o stringa.
Per l'inserimento di dati stringa in un comando INPUT non e' necessario siano compresi fra virgolette.
Rispondendo ad una richiesta di INPUT con dati diversi dal tipo di quelli richiesti, per esempio dati numerici quando era richiesta una stringa o viceversa, avremo come risultato la visualizzazione di un messaggio di errore:

?REDO FROM START

e verra' in pratica riproposto l' inserimento di dati.
Rispondendo all' INPUT invece con piu' dati di quelli richiesti avremo invece il messaggio di errore:

?EXTRA IGNORED

Un numero inferiore di dati di quello richiesti causera' la visualizzazione di un messaggio composto da due punti interrogativi (??) per segnalare che i dati inseriti sono insufficienti a consentire la ripresa dell' esecuzione del programma.

ESEMPIO:

```
10 INPUT X
20 PRINT X"IL QUADRATO E'" X^2
```

```
30 END

RUN

        ? 5
```

Questo 5 e' inserito dall' utente in risposta al punto interrogativo.

```
5 IL QUADRATO E' 25

READY.
```

INPUT#

FORMATO: INPUT# (numero del file), (lista di variabili)

FUNZIONE: Legge dati da un file disco sequenziale o random e li assegna a variabili del programma.

AZIONE: Il NUMERO DEL FILE e' il numero che era stato usato quando il file era stato aperto per l' inserimento dati.
La LISTA DELLE VARIABILI contiene i nomi delle variabili che saranno assegnati ai dati nel file.
Per gli esempi, dato che questicomportano comunque una serie di operazioni e di concetti riguardanti le periferiche si rimanda alla specifica sezione di questa Guida.

KEY

FORMATO: KEY [numero tasto, stringa]

FUNZIONE: Programma i tasti funzione o ne controlla i parametri assegnati.

AZIONE: A differenza di quanto avveniva nel CBM64 il C 128 permette di utilizzare pienamente ed in modo dinamico i tasti funzione programmandoli secondo le esigenze dell' utente. Questa possibilita' si rivela di particolare utilita' quando si scrivono dei programmi.
Infatti programmando i tasti funzione con comandi frequentemente usati si risparmia tempo e fatica e si evitano errori.
I quattro tasti funzione presenti possono definire 8 funzioni (da F1 a F8) di cui 4 se usati direttamente (numeri dispari) e 4 se usati indirettamente (numeri pari)
All' accensione del sistema sono assegnate le seguenti funzioni, che difiniremo quindi come standard, ai singoli tasti:

F1 = GRAPHIC abilita uno dei modi grafici quando si assegni un parametro e si prema poi il RETURN. Vedi il comando GRAPHIC per un maggior dettaglio.

F2 = DLOAD" scrive DLOAD" sullo schermo per cui tutto quello che e' necessario di fare e completare l' istruzione scrivendo il nome del programma e chiudendo le virgolette. Dopo il RETURN il programma sara' caricato da disco.

F3 = DIRECTORY lista tutti i files presenti su disco senza interferire con il programma in memoria.

F4 = SCNCLR esegue un clear di schermo

F5 = DSAVE" scrive DSAVE" sullo schermo e si comporta in proposito come per F2 in questo caso salvando invece che caricando il programma specificato.

F6 = RUN fa girare, dopo il return il programma presente in quel momento in memoria.

F7 = LIST esegue il listato del programma in memoria dopo il RETURN

F8 = MONITOR dopo il solito Return salta al programma MONITOR.

Naturalmente per tutti questi comandi e' necessario rivedere accuratamente il significato.
Digitando KEY e RETURN senza alcun parametro verranno visualizzate le funzioni assegnate a ciascun tasto. Queste funzioni possono essere variate e si puo' assegnare ad ogni tasto una funzione diversa impiegando appropriatamente il secondo parametro.
Notare che normalmente il listato viene eseguito su schermo, ma come sempre, anche in questo caso puo' essere utilmente impiegato anche il comando CMD per indirizzare l' uscita su altra periferica, ad esempio per avere su stampante l' elenco dei tasti funzione. Vedi indietro CMD. Il secondo parametro puo' essere non solo un comando ma anche una stringa alfabetica fra virgolette, una serie di dati, un comando, come del resto abbiamo gia' visto oppure una serie di comandi. La concatenazione fra dati, comandi, stringhe deve essere fatta con l' operatore + (piu') tenendo presenteche per ogni tasto non possono essere assegnati piu' di 246 caratteri complessivamente.
Il numero di caratteri e' piu' che sugfficiente

per gli scopi pratici di questo comando in quanto non si programmano lunghe sequenze di comandi per i tasti funzione.
Dopo aver programmato i tasti funzione e vedremo fra breve alcune applicazioni pratiche, questi possono essere usati sia in modo diretto che in modo programma.
Nel modo diretto, la stringa che si e' programmata per quel tasto sara' riportata sullo schermo se siamo in modo testo, altrimenti sara' invisibile se siamo nei modi grafici.
Premendo poi il tasto Return la stringa viene eseguita come un comando diretto o, se immessi, come una serie di comandi se separati dai due punti (:).
Se come ultimo carattere viene incluso un Return (CHR$(13)) sara' sufficiente premere il tasto funzione per eseguire il o i comandi relativi.
Nel modo programma si possono utilizzare i tasti funzione in risposta ai comandi INPUT o GET per riempire le variabili stringa nei programmi.
Per il comando INPUT, premendo un tasto funzione al quale siano assegnati valori appropriati, si visualizza la sua definizione subito dopo il punto interrogativo di richiesta informazioni tipico di questo comando.
Quando si preme il Return, la stringa viene trasferita alla variabile del comando INPUT.
Come abbiamo gia' detto in precedenza se come ultimo carattere della stringa e' incluso un Return con CHR$(13) sara' sufficiente premere il tasto funzione per trasferire la stringa contenuta nel tasto funzione alla variabile.
Anche la redifinizione dei tasti funzione oltre che ail loro impiegao puo' essere fatto sia in modo diretto che in modo programma.
Per riportare TUTTI i tasti funzione al valore originale basta premere il tasto di RESET del computer. Ricordiamo inoltre che all' atto del reset, dato anche con l' opportuno SYS i valori

dei tasti funzione vengono reimpostati su quelli originali:

ESEMPI:

KEY 1, "RUN" + CHR$(13)   Premendo il tasto F1 faremo girare il programma in memoria senza digitare RUN e poi Return che e' gia' incluso nel comando.

KEY 4,"GRAPHICO" + CHR$(13) + "LIST" + CHR$(13)
Premendo F4 si ritorna in modo grafico normale o standard e si lista il programma in memoria.

LET

FORMATO: LET (variabile)=(espressiono)

FUNZIONE: Assegna il valore di un' espressione ad una variabile.

AZIONE: Ricordiamo che nel BASIC utilizzato dai computers COMMODORE il comando LET e' opzionale perche' e sufficiente mettere il simbolo uguale (=) fra l' espressione e la variabile.

ESEMPIO:

```
110 LET D=12
120 LET E=12*2
130 LET F=12*4
140 LET SUM=D+E+F
```

puo' essere scritto:

```
110 D=12
120 E=12*2
```

```
130 F=12*4
140 SUM=D+E+F
```

LIST

FORMATO: LIST (numero di linea)

FORMATO: LIST (numero di linea)-(numero di linea)

FUNZIONE: Lista tutto o parte di un programma in memoria su una periferica attiva.

AZIONE: Questo comando puo' essere dato sia in modo diretto che in modo programma, ma comunque il BASIC ritornera' sempre in modo comando dopo la sua esecuzione.
Se il numero di linea da listare e' omesso allora il programma sara' listato iniziando dal numero di linea piu' basso, cioe' dall' inizio. In questo caso il listato puo' avere termine o perche' e' stato listato tutto il programma oppure perche' si preme il tasto di RUN/STOP. Se viene dato il numero di linea come parametro allora sara' listata solo quella linea.
Il secondo formato del comando consente le seguenti opzioni:

1- Se e' specificato solo il primo numero (sempre seguito dal segno meno (-)), allora sara' eseguito il listato da quel numero fino alla fine del programma o fino a quando il LIST non venga interrotto.

2- Se e' specificato solo il secondo numero, cioe' con un meno davanti, allora saranno listate tutte le linee dall' inizio del programma fino alla linea che segue il meno.

3- Se sono specificati entrambi i NUMERI DI LINEA allora la lista verra' fatta in quell' intervallo.

NOTA

Sul C 128 il procedimento puo' essere rallentato premendo il tasto con il simbolo della COMMODORE, fermato completamente con il RUN/STOP oppure fermato temporaneamente con il tasto NO SCROLL o con CONTROL S.
La pausa temporanea puo' essere disattivata con la pressione di un qualsiasi altro tasto.

ESEMPI:

LIST Lista il programma in memoria

LIST 500 Lista la linea 500

LIST 150- Lista tutte le linee da 150 alla fine.

LIST -1000 Lista tutte le linee del programma dall' inizio fino alla linea 1000 inclusa

LIST 150-1000 Lista le linee da 150 a 1000 incluse.

LOAD

LOAD "(nome del file)"[,n. periferica] [,flag di caricamento]

FUNZIONE: Carica un file dall' esterno in memoria.

AZIONE: Il NOME DEL FILE e' il nome che era stato usato quando il file era stato caricato su periferica con il comando SAVE.
Il comando LOAD chiude tutti i files aperti e cancella tutte le variabili e tutte le linee di programma presenti in quel momento in memoria prima di eseguire il caricamento di quel dato programma.
Se il comando LOAD e' eseguito da programma, allora il programma caricato gira dopo il LOAD e tutti i files di dati aperti restano aperti.
In questo caso il LOAD puo' essere utilizzato per concatenare piu' programmi o segmenti dello stesso programma. Naturalmente nessuna delle variabili viene azzerata durante l' operazione di concatenamento.
Quando il disco, periferica 8, e' specificato, il nome del file deve essere preceduto dal numero del drive e dai due punti (vedi l' esempio seguente). Se nessun numero di drive e' dichiarato e sono connessi 2 drives, allora la ricerca avvera' su entrambi i drives.
L' ultimo parametro del comando che puo' assumere valori di 0 o di 1 determina in quale parte della memoria deve essere caricato il programma.
Un valore di 0, che e' anche il valore di DEFAULT, indica al computer di caricare il programma all' inizio dell' area di programma BASIC.
Un valore 1 indica al C 128 di caricare il programma nella stessa area in cui il programma era quando fu salvato. Generalmente si usa per i programmi in Linguaggio Macchina.

ESEMPI:

LOAD"0:PIPPO",8 Il file PIPPO viene caricato dall' unita' 0 del drive 8.

LOAD"*",8 Carica il primo file programma presente su disco.

LOAD Carica il primo programma che trova sull' unita' a cassetta.

LOAD (B$),8 Carica da disco il programma il cui nome e' indicato nella variabile B$.

LOAD"DEMO",8,1 Carica da disco un programma di nome DEMO e immettilo nelle locazione di mmoria in cui era quando fu salvato.

LOCATE

FORMATO: LOCATE X,Y

FUNZIONE: Colloca il PC in uno scelto punto dello schermo.

AZIONE: E' giunto il momento di spiegare cosa sia il PC e di comprenderne i concetti fondamentali anche se rimandiamo alla sezione dedicata alla grafica informazioni piu' approfondite.
Il PC, abbreviazione di PIXEL CURSOR e' simile al cursore mobile, cioe' quel quadratino lampeggiante che si puo' osservare nel modo 40 colonne, che indica dove apparira' il prossimo carattere.
Nei modi grafici selezionabili tramite il comando GRAPHIC, il PC che pero' resta invisibile, indica dove verra' collocato il prossimo punto sullo schermo sia in Alta risoluzione che in MULTI-COLOR.
Nei comandi grafici che in parte abbiamo visto, nei casi in cui mancano le coordinate opzionali

il PC e' usato come coordinate di DEFAULT.
Il comando LOCATE permette di spostare il PC in un punto qualsiasi dello schermo che in alta risoluzione e' una finestra di 320x200 punti.
Naturalmente i risultati del comando LOCATE non saranno visibili fino a quando non verra' effettivamente disegnato qualcosa con i comandi BOX, DRAW, CIRCLE o riempito uno spazio con PAINT.
La X del comando rappresenta la coordinata orizzontale o sull' asse X dello schermo espressa in numero di punti. Lo schermo visibile su questa coordinata puo' avere un' intervallo da 0 a 320. Quando x= 0 il PC e' posizionato sul margine sinistro dello schermo stesso.
La Y del comando rappresenta la coordinata verticale o sull' asse Y dello schermo espressa in numero di punti. Lo schermo visbile su questa coordinata puo' avere un intervallo fra 0 e 200. Quando Y = 0 il PC e' sul margine superiore dello schermo.
Le coordinate X e Y possono essere espresse sia come valore assoluto che come scarto dalla presente posizione del PC. Facendo precedere il PC da un segno positivo (+) o negativo (-) il PC viene mosso in una direzione positiva o negativa relativamente alla sua posizione attuale.
In altre parole il segno + prima del valore X muove il PC verso destra dalla posizione attuale, mentre il segno meno lo muove verso sinistra.
Nello stesso modo il segno + davanti al valore Y muove il PC verso il basso rispetto alla posizione di quel momento mentre il segno - lo muove verso l' alto.

ESEMPI

LOCATE 320,200 Pone il PC nell' angolo in basso a destra dello schermo

LOCATE 0,0 Pone il PC nell' angolo in alto a sinistra.

LOCATE 160,100 Pone il PC esattamente nel centro dello schermo.

LOCATE -20,+30 Muove il PC di 20 punti a sinistra e di 30 in basso.

NOTA

Il valore attuale di PC, cioe' il valore delle sue coordinate puo' essere trovato con la funzione RDOT (n).
Quando ad N si assegna il valore 0 viene visualizzata la coordinata X del PC. Quando si assegna il valore 1 viene visualizzata la coordinata Y del PC, mentre quando si asegna un valore 2 verra' visualizzato il valore corrispondente al colore sorgente relativo al PC. (Per il colore vedi anche il comando COLOR).
LOCATE puo' essere utilizzato in forma diretta o in modo programma.

MONITOR

FORMATO: MONITOR

FUNZIONE: Attiva il programma MONITOR

AZIONE: Per questa funzione e per i comandi che mette a disposizione vedi la sezione dedicata al Linguaggio Macchina.

MOVSPR

FORMATO:

1) MOVSPR n,x,y

2) MOVSPR n, +/-x, +/-y

3 MOVSPR n,x;y

4) MOVSPR n,[angolo x][velocita' #y]

FUNZIONE: Posiziona o muove gli Sprites.

AZIONE:Vediamo i parametri:

N E' il numero dello Sprite (1-8)

X, Y Sono le coordinate dello Sprite

ANGOLO Angolo del movimento in senso orario relativo alle coordinate originali (0-360)

VELOCITA' Rapidita' con la quale viene mosso lo Sprite (0-15)

Caratteristica particolare del C 128 e' la capacita' di posizionare ed animare gli Sprites. Per ogni Sprite attivato e' possibile usare il comando MOVSPR per regolarne la posizione e farlo muovere.
Come abbiamo visto N che e' costante in tutte le forme del comando indica lo Sprite sul quale si vuole agire. Avremo poi delle coordinate, che non sono le coordinate viste per i comandi grafici, ma che spiegheremo in seguito e che indicano da dove parte il movimento. In altre parole la locazione dello Sprite.
Anche in questo caso alle coordinate si possono dare valori assoluti o valori relativi o scarti

relativi. E' tuttavia importante ricordare che utilizzando degli scarti relativi per posizionare lo Sprite bisogna fare attenzione perche' la nuova posizione viene calcolata dalla posizione corrente dello sprite piuttosto che dalla locazione attuale del PC.
Gli Sprites vengono posizionati rispetto al loro angolo superiore sinistro. Vi e' infatti una posizione specifica dello schermo chiamata FINESTRA in cui sono visibili gli Sprites e questa posizione e' esattamente data dalle coordinate che definiscono i limiti dello schermo nel modo ALTA-RISOLUZIONE. Vediamo alcuni esempi:

ESEMPI

MOVSPR 2,100,160 Posiziona lo Sprite 2 alle coordinate date che sono all' incirca il centro dello schermo.

MOVSPR 1,90 #7 Muove lo Sprite 1 con un angolo di 90 gradi. Il movimento avviene a velocita' media cioe' 7.

NEW

FORMATO: NEW

FUNZIONE: Cancella il programma attualmente in memoria e azzera tutte le variabili.

AZIONE: Questo comando puo' essere eseguito in forma diretta o in forma programma. Il Basic ritorna sempre a livello comando dopo l' esecuzione del NEW.

ON...GOSUB e ON...GOTO

FORMATO: ON(espressione)GOTO(lista di numeri di linea)
ON(espressione)GOSUB(lista di numeri di linea)

FUNZIONE: Salta ad una delle diverse linee specificate come parametri di conseguenza alla risoluzione di un' espressione.

AZIONE: Il valore dell' espressione determina quale numero di linea nella lista sara' utilizzato per il salto. Se per esempio il valore e' 3, allora la destinazione del salto sara' il terzo numero di linea nella lista.
Se il valore e' un non intero, la parte frazionaria sara' arrotandata verso il numero inferiore.
Nel comando ON...GOSUB, ogni numero di linea nella lista deve corrispondere al primo numero di linea della subroutine alla quale si rimanda.
Se il valore dell' espressione e' negativo verra' visualizzato un messaggio di:

ILLEGAL QUANTITY

Se il valore dell' espressione e' zero o maggiore del numero di dati presenti nella lista, allora il controllo passa al comando o alla linea seguente.

ESEMPIO:

```
100 ON L-1 GOTO 150,300,320,390
```

OPEN

FORMATO: OPEN (numero di file logico) [,(numero di periferica) [,(indirizzo secondario) [,"(nome del file)(parametri)]]]
FUNZIONE: Rende disponibile (apre) un canale di I/O su una periferica.

AZIONE: Il numero di file logico DEVE essere specificato e deve essere compreso nell' intervallo fra 1 e 255. Se manca il numero di periferica questi viene sottinteso uguale a 1 che corrisponde alla prima cassetta. La mancanza di indirizzzo secondario e di nome del file non viene eguagliata a niente.
Numeri di files logici maggiori di 128 inviano un ritorno carrello ed un LINE FEED con ogni PRINT#, mentre i numeri di files logici minori di 128 inviano solo un ritorno carrello. I ritorno carrello possono essere soppressi con un punto e virgola (;).
Un file viene scritto come programma (PRG) se non e' specificato che si tratta di File Sequenziale (S). I files sequenziali sono aperti per essere letti se il simbolo W (per Write) non e' specificato.
Il numero del drive deve essere specificato (cioe' deve essere detto se 1 o 0) se trattasi di unita' modello 2040. Per gli altri modelli, in mancanza di numero di drive viene preso come uguale a 0.
I files possono essere aperti su nastro (periferica 1 o 2), disco (periferica 8), stampante CBM (periferica 4) o schermo (periferica 3).

ESEMPIO:

```
10 OPEN 2,8,2,"0:DATAFILE,S,W"
20 FOR I=1TO10
```

```
30 A$=CHR$(I)
40 PRINT#1,A$
50 NEXT I
60 CLOSE 2
```

PAINT

FORMATO: PAINT [colore sorgente],X,Y [,modo]

FUNZIONE: Colora un' area.

AZIONE: Vediamo i parametri:

COLORE SORGENTE Determina la sorgente di colore da utilizzare. I valori da assegnare a questo parametro sono:

0 = Sfondo
1 = Primo piano
2 = Multi-color 1
3 = Multi-color 2

X, Y Sono le coordinate da cui inizia la colorazione. Il valore di DEFAULT e' al solito quello del PC.

MODO Specifica che tipo di delimitazione fermera' il colore.

La colorazione infatti iniziando dalle coordinate come detto in precedenza continua intorno alla posizione da esse indicate fino a quando non incontra una delimitazione. Se il valore del parametro MODO = 0 PAINT espandera' il colore sino a quando incontra una delimitazione che abbia il suo stesso colore sorgente.

Se invece MODE =1 PAINT colorera' fino ad incontrare un qualsiasi contorno di primo piano o di Multi-color.
La posizione finale del PC sara' a X,Y quando la colorazione e' completata. Se la coordinata determinata da X,Y si trova su un punto della stessa sorgente di colore utilizzata con il comando PAINT, allora non sara' effettuata nessuna colorazione.

ESEMPIO

```
100 COLOR 1,11:
110 GRAPHIC 1,1
120 CIRCLE1,80,100,50,40
130 CIRCLE2,100,100,50,40
140 PAINT 1,110,100,1
150 PAINT 2,110,100,0
160 PAINT 0,110,100
```

Nell' esempio le linee da 100 a 130 servono a definire il modo grafico, i colori ed a disegnare dei cerchi.
La riga di programma o linea 140 riempira' le porzioni coincidenti dei due cerchi con l' attuale colore di primo piano.
La linea 150 riempira' tutto il cerchio tracciato dalla linea 130.
Usando una sorgente di colore con valore 0 (colore di sfondo) viene effettivamente cancellata un' area ed il suo contorno appunto con l' esecuzione della linea 160.

PLAY

FORMATO: PLAY "Vn,On,Tn,Un,Xn,note, durata"

AZIONE: Vediamo innanzi tutto i numerosi parametri che caratterizzano questo comando.

Vn = Voce (n=1-3)

On = Ottava (n=0-6)

TN = Inviluppo di TUNE (n=0-9) in cui:

0 = Piano
1 = Fisarmonica
2 = Calliope
3 = Tamburo
4 = Flauto
5 = Chitarra
6 = Clavicembalo
7 = Organo
8 = Tromba
9 = Xilofono

UN = Volume (n=0-15)

Xn = Filtro attivo (n=1) disattivo (n=0)

NOTE A,B,C,D,E,F,G le note musicali secondo la notazione americana.

DURATA delle note e valori:

\# Diesis
$ Bemolle
W Intero
H Mezza
Q Quarto
I Ottavo
S Sedicesimo
. Puntata
R Pausa
M Attesa per tutte le voci che stanno

suonando.

Tutti i parametri durata, eccetto che M e R precedono le note musicali nella stringa.

Con questo comando si puo' comporre della musica usando caratteri stringa. La musica potra' essere eseguita introducendo questi caratteri da tastiera in modo diretto o includendoli in stringhe del programma prededute e quindi fatte suonare con PLAY. Come abbiamo visto le note vengono definite con le lettere dalla A alla G che corrispondono alla scala dal LA al SOL. La durata viene dichiarata dalle lettere W,H,I,S ed ogni nota che segue una delle lettere di durata viene suonata alla stessa lunghezza fin quando la durata stessa non viene cambiata. La lettera R indica la pausa della durata di una nota.
Le note che vengono precedute dal segno # sono suonate come diesis quelle da $ come bemolli. Le note precedute da un . (punto) durano una volta e mezzo le normali.

ESEMPI

PLAY"V2O5T3U8X0ABCDEFG" Suona le note ABCDEF, una scala, selezionando la voce 2, la quinta ottava, con strumento tamburo al volume 8 con il filtro disinserito.

PLAY"T0U10$CDEFGAB" Suona le note CDEFGAB con i valori di DEFAULT per quanto riguarda la voce e l' ottava scelta, usando un pianoforte, a volume 10 in bemolle.

POKE

FORMATO: POKE I,J

FUNZIONE: Scrive un byte in una locazione di memoria.

AZIONE: Questo comando consente di cambiare un qualsiasi valore presente nella memoria RAM del C 128 e molti dei valori dei suoi registri.
Relativamente ai banchi di memoria questo comando opera sul banco selezionato inquel momento.
I, che deve essere un' espressione intera, e' la locazione di memoria, mentre J, anche questo deve essere un' intero, e' il dato che deve essere inserito.
J deve essere un valore compreso nell' intervallo fra 0 e 255, mentre I deve essere compreso nell' intervallo fra 0 e 65535.
POKE puo' essere usato in modo diretto o in modo programma.
La funzione complementare di POKE e' PEEK che legge il contenuto di una locazione di memoria.

ESEMPIO:

```
10 POKE 53280,2
```

Trovandosi nel modo 40 colonne cambia il colore del bordo.

```
10 POKE 53281,10
```

Come nell' esempio precedente per il colore di fondo.

NOTA

Ricordiamo che ci sono delle locazioni di memoria, la parte ROM, che non posono essere scritte, mentre su altre, in particolare quelle sulle pagine da zero a quattro, e' bene adoperare notevoli cautele a meno che non si posseggano conoscenze approfondite.

PRINT E PRINT#

FORMATO: PRINT (#(file logico),) ((lista di espressioni))

FUNZIONE: Uscita di dati su schermo o su un canale

AZIONE: Se e' omesso il parametro (lista di espressioni), viene stampata o visualizzata una linea bianca.
Se invece la lista di espressioni e' inclusa come parametro i valori sono stampati sul terminale scelto.
Le espressioni possono essere numeriche o stringa. Ricordiamo che le stringhe devono essere racchiuse fra virgolette.

POSIZIONI DI STAMPA

La posizione di ogni dato da stampare e' determinata dalla punteggiatura utilizzata per separare i dati nella lista stessa.
Il Basic divide le linee in zone di stampa di 10 spazi ciascuna. Se i dati nella lista sono

separati da una virgola allora il successivo valore verra' stampato nella successiva zona di stampa , cioe' dopo 10 caratteri.
Un punto e virgola invece fa stampare il dato immediatamente dopo il precedente.
L' inserimento di uno o piu' spazi fra i dati ha lo stesso effetto dell' inserimento di punti e virgola.
Se al termine della lista di espressioni viene messo una virgola o un punto e virgola il Sistema Operativo del computer "RICORDA" di far partire il primo comando PRINT che trovera' negli spazi relativi. Cioe' nella prossima zona (10 spazi dopo) oppure, con il punto e virgola, immediatamente accanto.
Se la lista di espressioni termina senza nessuno dei segni di punteggiatura detti allora al termine dei dati stampati avremo un ritorno carrello.
Se la linea da stampare e' piu' lunga di 80 caratteri o colonne, allora la stampa continuera' nella successiva linea fisica.
Un ritorno carrello viene comunque sempre inviato dopo ogni comando PRINT#.
Se il FILE LOGICO e' un numero piu' grande di 128 viene inviato alla periferica ANCHE un line feed.
Entrambe le funzioni associate al comando possono essere annullate dall' inserimento di un punto e virgola.
I numeri stampati sono sempre seguiti da uno spazio.
I numeri positivi sono preceduti da uno spazio mentre quelli negativi dal segno meno. Ed e' questa una particolarita' da tenere presente per evitare, in particolare nelle stampe miste di dati numerici ed alfanumerici insieme, errori di allineamento.
Invece del comando PRINT si puo' usare la forma abbreviata che e' un punto esclamativo.

ESEMPIO:

```
10 X=5
20 PRINT X,X+5,X-5,X*(-5)

RUN

5         10          0         -25

READY.
```

In questo esempio le virgole utilizzate nel comando PRINT consntono che ogni valore sia stampato all' inzio della succesiva zona di stampa.

ESEMPIO:

```
10 INPUT X
20 PRINTX "IL QUADRATO E'"X^2"E";
30 PRINT"IL CUBO E'" X^3"
40 PRINT
50 GOTO 10

RUN

? 9

9 IL QUADRATO E' 81 E IL CUBO 729

?21
21 IL QUADRATO E' 441 E IL CUBO E' 9261
```

In questo esempio il punto e virgola alla fine della linea 20 fa si che entrambi i risultati dell' azione dei comandi PRINT siano stampati sulla stessa linea, mentre il comando PRINT a vuoto della linea 40 fa si che una linea bianca sia stampata prima della richiesta di ingresso

dati conseguente al GOTO.
La versione del comando PRINT# che serve ad inviare i dati ad una periferica sara' vista, anche come esempi applicativi nel piu' completo discorso nella sezione delle periferiche.

PRINT USING

FORMATO: PRINT[#n.file] USING listato del formato"; lista da stampare

FUNZIONE: Uscita dati secondo un formato prestabilito

AZIONE: L' insieme di istruzioni contenute in questo comando permettono di definire il formato della stringa e degli elementi numerici che si vuole visualizzare sullo schermo, , stampare sulla stampante o inviare ad altro dispositivo in output.
Il formato desiderato deve essere racchiuso fra virgolette e sara' il listato. Dovremo quindi aggiungere un punto e virgola (;) ed una lista di cio' che si vuole stampare nel formato del listato di stampa. La lista puo' contenere sia delle variabili che delle costanti, intendendo quest' ultime come gli effettivi valori che si vuole stampare.
Facciamo subito un esempio:

```
5  X=32: Y=100.23: A$="CAT"
10 PRINT USING "$##.##";13.25,X,Y
20 PRINT USING "###>#";"CBM",A$
```

Quando viene fatto girare questo programma visualizzera':

```
$13.25    $32.00    $*****

          CBM     CAT
```

Spiegheremo subito i motivi, per il momento notare che la stampa degli asterischi invece del valore di Y avviene perche' il valore di Y e' di 5 cifre e non e' conforme al formato di stampa scelto.
Nella seconda riga notare che vengono lasciati degli spazi prima di stampare CBM perche' cio' e' quanto definito nel formato del listato.
Vediamo ora i segni utilizzabili con i tipi di dati da stampare:

CARATTERE	NUM	STRINGA
Diesis (#)	X	X
Segno piu' (+)	X	
Segno meno (-)	X	
Punto (.)	X	
Virgola (,)	X	
Dollaro ($)	X	
4 Frecce (^^^^)	X	
Uguale (=)	X	X
Maggiore (>)	X	X

Il simbolo diesis (#) riserva dello spazio per un singolo carattere nel campo di output. Se il dato contiene piu' caratteri di quanti siano presenti nel campo del formato allora l' intero campo viene riempito con asterischi e non viene quindi stampato alcun numero.

ESEMPIO:

```
10 PRINT USING "####";X
```

Per i seguenti valori assegnati ad X osservare cosa viene stampato accanto:

```
X = 12.34       12
X = 567.89      568
X = 9876543     ****
```

Quando ci troviamo con una stringa i dati della stessa vengono troncati ai bordi del campo. In altre parole verranno stampati SOLO tanti caratteri quanti sono i # nel formato. Naturalmente il troncamento avviene sulla destra.
I segni + e - possono essere usati nella prima o nell' ultima posizione dl campo del formato, ma non in tutte e due. Se il numero e' positivo verra' stampato o visualizzato il segno piu', se e' negativo il segno meno.
Se si utilizza il segno meno per un numero positivo verra' visualizzato uno spazio vuoto nella posizione del carattere indicata dal segno meno.
Se per un elemento di dati numerici non vengono utilizzati ne' il segno piu' ne' il segno meno nel campo del formato allora verra' stampato un segno meno davanti alla prima cifra o al simbolo del dollaro se il numero e' negativo. Se il numero e' positivo non verra' stampato alcun segno.
Cio' sta a significare che se il numero e' positivo si puo' stampare un' altro carattere.
Se vi sono troppi caratteri da immettere nel campo specificato dal simbolo # e dai segni + e - si incorre allora in un errore di sovradimensionamento ed e' per questo che il campo si riempe di asterischi come abbiamo visto.
Il punto (.) che corrisponde alla nostra virgola decimale indica infatti la posizione della virgola decimale in un numero. Si puo' avere una sola virgola decimale in un qualsiasi campo. Se il punto non viene specificato avremo allora un

arrotondamento all' intero piu' vicino che verra' quindi stampato senza posizioni decimali.
Quando si dichiara la presenza di una virgola decimale il numero di cifre che presiede la virgola decimale, incluso il segno meno se il valore e' negativo, non deve superare il numero di # presenti prima della virgola decimale. Se le cifre sono troppe accade quanto gia' osservato in precedenza ed invece di aver stampato un valore ci troveremo con il campo pieno di asterischi ad indicare il nostro errore.
Una virgola consente l' immissione di virgole nei campi numerici. Infatti la posizione della virgola nel formato indica la posizione della virgola stessa nel formato di stampa che poi avremo.
Ricordiamo che verranno stampate SOLO le virgole all' interno del numero, mentre le virgole non utilizzabili vengono stmpate alla sinistra della prima cifra in qualita' di carattere di riempimento. Inoltre almeno un carattere # deve precedere la prima virgola nel campo.
Se si dichiarano le virgole in un campo ed il numero che ne viene poi fuori e' un numero negativo, verra' stampato un segno meno come primo carattere anche se la posizione del carattere e' specificata come virgola.
In altre parole si avra' uno spostamento del campo verso destra.

ESEMPI:

	CAMPO	ESPRESSIONE	RISULTATO
1)	##.#+	-.01	0.01-
2)	##.#-	1	1.0
3)	####	- 100.5	- 101

4) #### - 1000 ****

5) ###. 10 10.

6) #$## 1 $1

NOTE

Nel caso 1) si ha l' aggiunta di uno ZERO all' inizio del campo mentre nel caso 2) si ha alla fine dello stesso campo di stampa.
I quattro # nel caso 3) fanno eseguire un' arrotondamento senza decimali, mentre nel caso 4) generano, come gia' detto un OVERFLOW. Nel caso 5) avremo la giunta di una virgola decimale, mentre nell' ultimo caso avremo la stampa, abbastanza inutile per noi, del segno del dollaro ($), perche' infatti inserendo un simbolo $ questi verra' stampato nel numrero stesso.
Se si desidera che il simbolo del $ si posizioni automaticamente, cioe' venga stampato sempre prima del numero e' necessario dare all' espressione che segue il comando almeno un simbolo # prima del segno dollaro.
Infatti se si dichiara un segno $ senza un # iniziale, allora il segno $ sara' stampato nella posizione data nel campo del formato contenente un simbolo $, allora il programma stampera' una virgola o un segno prima del simbolo stesso.
Il simbolo delle quattro freccette verso l' alto serve ad informare il sistema che il numero deve essere stampato in formato esponenziale o notazione scientifica (E+).Sara' anche necessario aggiungere un simbolo # alle 4 freccette per specificare la larghezza del campo di stampa.
Le quattro freccette possono essere immesse sia

prima che dopo il segno #.
In altre parole la sintassi di stampa di numeri in notazione esponenziale e' la seguente:

1) I simboli delle freccette servono ad indicare la notazione scientifica.

2)Se si immettono le freccette in numero maggiore di uno ma minore di 4 verra' visualizzato un messaggio di SYNTAX ERROR

3)Se si specificano piu' di quattro freccette verranno utilizzate solo le prime 4 e dalla quinta in poi verranno ritenute dal sistema come simboli di NON-TESTO.

Il segno = (uguale) e' utilizzato per centrare una stringa in un campo e la larghezza del campo viene specificata dal numero di caratteri (# e =) nel campo del formato.
Se la stringa contiene iun numero di caratteri inferiore alla larghezza del campo la stringa verra' centrata nel campo stesso.
Se invece la stringa presenta piu' caratteri di quanti il campo prebvisto ne metta a disposizione allora verranno tagliati i caratteri a cominciare dalla destra.
Il segno maggiore (>) viene utilizzato per allineare a destra una stringa.

PUDEF

FORMATO: PUDEF (nnnn)

FUNZIONE: Ridefinisce i simboli nel comando PRINT USING

AZIONE: PUDEF seguito dal parametro n che e' una combinazione di un massimo di 4 caratteri, permette di ridefinire fino a 4 simboli nell' istruzione PRINT USING.
E' possibile cambiare spazi vuoti, virgole, virgole decimali (ricordiamo che e'n il punto) e simboli $ in altri caratteri sostituendo il nuovo carattere nella corretta posizione della stringa di controlo PUDEF. Vediamo cosa significano queste posizioni.

La prima posizione e' il carattere di riempimento. In presenza di DEFAULT avremo un spazio vuoto. Inserire qui un nuovo carattere quando si vuole che un altro carattere appaia al posto degli spazi vuoti.

La seconda posizione e' la virgola per cui il valore di DEFAULT sara' la virgola.

La terza posizione e' il punto o virgola decimale. Per DEFAULT punto.

La quarta posizione e' il simbolo del dollaro. Per DEFAULT il dollaro.

ESEMPI
10 PUDEF "*" Stampa cioe' un asterisco al posto degli spazi vuoti.

10 PUDEF ".," Stampa punti decimali al posto delle virgole e virgole al posto di punti decimali.

READ

FORMATO:READ (lista di variabili)

FUNZIONE: Serve per leggere i valori dai comandi DATA ed assegnarli alle variabili.

AZIONE: Il comando READ deve essere sempre usato insieme ad uno o piu' comandi DATA.
Le variabili lette dal comando READ possono essere numeriche o stringa, ma il valore letto deve essere del tipo della variabile specificata.
Se infatti si tenta di leggere dal DATA una stringa e si era assegnato per quella lettura una variabile numerica o intera avremo una segnalazione di SYNTAX ERROR.
Un singolo comando READ puo' leggere, naturalmente nell' ordine in cui sono presenti, uno o piu'comandi DATA oppure piu' comandi READ possono leggere lo stesso comando DATA.
Se il numero delle variabili nella " LISTA DELLE VARIABILI" supera il numero di elementi del comando o dei comandi DATA avremo un errore di:

OUT OF DATA.

Se invece il numero delle variabili di READ e' inferiore al numero di elementi presenti nel comando DATA, allora il successivo READ iniziera' la lettura dal primo elemento del DATA non letto.
Se non ci sono altri comandi READ allora i restanti valori dei DATA verranno ignorati.

Si ricorda che per rileggere i valori dei DATA dall' inizio e' necessario fare uso del comando RESTORE.

ESEMPIO:

```
80  FORI=1 TO 10
```

```
90  READ A(I)
100 NEXT
110 DATA 3.08,519,3.12,3.89,4.24
120 DATA 5.08,5.55,4.00,3.16,3.37
.
.
.
```

Questo pezzo di programma legge con READ i valori dai comandi DATA e li immette nella matrice definita alla linea 90 cioe' A(I). Dopo l' esecuzione il valore di A(1) sara' 3.08, quello di A(2) sara' 5.19 e cosi' via.

ESEMPIO:

```
10 PRINT "NOME","CAP", "CITTA'"
20 READ N$,C,C$
30 DATA "ROSSI G.",20133,"MILANO"
40 PRINT N$,C,C$

RUN

NOME      CAP      CITTA'

ROSSI     20133    MILANO

READY.
```

Questo programma legge con il comando READ sia valori numerici che stringa presenti nel DATA della linea 30.

RECORD

FORMATO: RECORD# (file logico), (record)[,(byte)]

FUNZIONE:E' utilizzato prima di GET#, INPUT# PRINT# per posizionare il puntatore ad un carattere del record in un file ad accesso casuale.

AZIONE: Il numero del record deve essere in un' intervallo fra 0 e 65535, mentre il BYTE POINTER deve esere fra 1 e 254. In mancanza di valore verra' assegnato il valore 1 al BYTE POINTER.
Quando il puntatore dl record e' fissato ad un valore piu' alto del numero dell' ultimo record del File viene generata la seguente condizione: se e' usata l' istruzione PRINT# il file viene espanso fino al record richiesto dal puntatore.
Se invece e' utilizzata l' istruzione INPUT# allora viene restituita una stringa nulla e viene fissata la variabile ST con il valore di EOI e la variabile DS$ conterra' "RECORD NOT PRESENT".
Quando e' utilizzata una variabile per il cui valore e' l' indirizzo del record deve essere messa fra parentesi.

ESEMPIO:

```
10 DOPEN #1,"RANDOM",L80
20 FORI=1 TO 10
30 RECORD#1,(I)
40 PRINT#1,"ABCDEFG"
50 NEXT
60 DCLOSE#1
```

REM

FORMATO: REM (note)

FUNZIONE: Consente di inserire delle note esplicative all' interno di un programma.

AZIONE: Il comando REM non viene eseguito ma e' visualizzato esattamente come e' stato inserito, eseguendo il LIST del programma.
Si puo' eseguire un salto ad una riga di programma contenente un REM tramite un comando GOTO e GOSUB. In questo caso verra' eseguita la prima linea o il primo comando eseguibile deopo il REM stesso.
Una nota (REM) non puo' essere seguita da un altro comando sulla stessa linea a meno che non sia separato dai due punti.

ESEMPIO:

```
120 REM CALCOLO VELOCITA' MEDIA
130 FOR I= 1 TO 20
140 SU= SU+V(I)
.
.
.
```

RENAME

FORMATO: RENAME [D (x),] "(vecchio nome)" TO " (nuovo nome)" [ON U (y)]

FUNZIONE: Cambia il nome di un file disco

AZIONE: Il disco non eseguira' il comando RENAME su un file attualmente aperto.
In mancanza di specificazioni di drive e/o di

unita' verranno selezionati rispettivamente il drive 0 e l' unita' 8.
Quando vengano usate variabili o espressioni da calcolare come nomi di file (vecchio e/o nuovo) queste devono essere racchiuse fra parentesi.

ESEMPIO:

RENAME "CAPITOLO 1" TO "PRAGRAFO 1"

RENUMBER

FORMATO: RENUMBER [nuovo n. della linea di partenza] [,incremento][,vecchio n. della linea di partenza]

FUNZIONE:Rinumera le linee di un programma Basic.

AZIONE: In questo comando che serve per rinumerare totalmente o parzialmente le linee di un programma Basic il nuovo numero della linea di partenza sara' il numero di linea DOPO la numerazione. L' incremento sara' l' intervallo fra i numeri di linea e di conseguenza il vecchio numero della linea di partenza sara' il punto di partenza del vecchio programma per la rinumerazione. In particolare quest' ultimo parametro consente di rinumerare anche parti di programma.
Questo comando puo' essere impiegato solo in modo diretto. I valori di DEFAULT nel caso cioe' che non si assegnino le linee di partenza e l' intervallo saranno 10,10.
Si puo' avere la segnalazione di due errori:

LINE NUMBER NOT FOUND ERROR quando non esiste il numero di linea al quale abbiamo tentato di

riferirsi.

OUT OF MEMORY quando si tenta di espandere il programma oltre i propri limiti.
Entrambi questi errori lasciano inalterato il programma di partenza.

ESEMPI

RENUMBER 100,10,10 Rinumera un programma che partiva dalla linea 10 iniziando ora dalla linea 100 con un incremento di 10 fra le linee.

RENUMBER Rinumera un programma impiegando i valori di DEFAULT, per cui il programma sara' rinumerato a partire dalla linea 10 con un incremento di 10.

RESTORE

FORMATO: RESTORE [linea#]

FUNZIONE: Consente che i contenuti dei comandi DATA siano riletti dall' inizio.

AZIONE: Dopo che e' stato eseguito un comando RESTORE il successivo comando READ riprendera' la lettura dei contenuti dei DATA dall' inizio e non dove era terminata la lettura del precedente comando READ.

ESEMPIO:

```
10 READ A,B,C
20 RESTORE
30 READ D,E,F
40 DATA 57,67,77
.
```

.
.

In questo caso al momento in cui vengono eseguite le linee 20 e 30 i valori del comando DATA vengono riletti da capo.

RESUME

FORMATO: RESUME [linea #/NEXT]

FUNZIONE: Esegue una nuova partenza del programma dopo che e' stato individuato un errore.

AZIONE: Questo comando consente di far proseguire l' esecuzione di un programma dopo che durante la sua esecuzione siamo incorsi in un errore.
Se non vengono specificati dei parametri RESUME prova a rieseguire la linea in cui e' contenuto l' errore.
RESUME NEXT continua l' esecuzione dall' istruzione successiva a quella in cui era contenuto l' errore.
RESUME riga # saltera' alla linea specificata dopo che e' stato incontrato un errore. In altre parole equivale ad un comando GOTO con la condizione di errore.
Puo' essere usato solo in modo programma.

ESEMPIO:

```
100  INPUT"SCRIVI UN NUMERO";B
110  TRAP1000:C=1/B
120  PRINT"IL RECIPROCO E'";C
130  INPUT"ANCORA (S/N)";A$:IF A$="S"THEN100
140  STOP
```

```
1000 INPUT"DAMMI UN NUMERO <> DA ZERO";B
1010 RESUME 110
```

RUN

FORMATO: RUN [(numero di linea)]

FUNZIONE: Esegue il programma attualmente in memoria.

AZIONE: Se viene specificato un numero di linea l' esecuzione di un programma iniziera' allora da quel numero. In caso contrario incomincera' dall' inizio.

Il programma si arresta e il computer ritorna nel modo comando quando:

- Non ci sono altri numeri di linea da eseguire

- Sono stati incontrati dei comandi STOP o END

- Il programma e' incorso in un errore durante l' esecuzione, salvo il caso di trattamento con le trappole di errore.

- E' stato incontrato un comando LIST

SAVE

FORMATO: SAVE ["(nome del file)" [,(numero della periferica) [,(COMANDO)]]]

FUNZIONE: Registra un file programma su nastro o disco.

AZIONE: Se manca il numero di periferica il file verra' registrato su cassetta (per la precisione sulla n. 1).
Se nel parametro (COMANDO) viene immesso uno 0 viene scritto un EOT cioe' un END OF TAPE (fine nastro) al termine della reistrazione del file.
Se invece il COMANDO e' diverso da 0 allora dopo la registrazione del file verra' messo un bloco di EOT.
E' possibile, ma solo per le registrazioni su cassetta NON usare un nome di file.
Quando invece viene utilizzato per il nome del file un' espressione da calcolare o una variabile, questa deve essere racchiusa fra parentesi.

Per registrare dati su disco e' necessario specificare il numero della periferica.
Se si desidera specificare il drive questi dovra' precedere il nome del file (vedi esempio sotto) ed essere da questi separato da due punti.
Se non e' specificato il numero del drive il computer registrera' (o provera' a registrare) sull' ultimo drive al quale abbiamo avuto accesso.

ESEMPIO:

SAVE

Ricordiamo che in questa forma avremo una risposta di PRESS PLAY AND RECORD per eseguire una registrazione su cassetta.

SAVE"PIPPO"

SAVE"",2

Salvataggio sulla cassetta numero 2

SAVE"1: PIPPO",8

Salvataggio di un file di nome PIPPO sul drive 1 dell' unita' 8.

SCALE

FORMATO: SCALE n [,X max,Y max]

FUNZIONE: Varia la scala del disegno.

AZIONE:Nei modi grafici, cioe' nei modi Multicolor e Alta risoluzione, si puo' variare la scala degli assi con il comando SCALE. Con Scale 1 si attiva la modalita di variazione.
I parametri opzionali possono assumere valori diversi a secondo di che modo grafico abbiamo attivato.

ALTA RISOLUZIONE

X puo' variare da 320 a 65535. Il valore dei DEFAULT e' 1023.

Y puo' variare da 200 a 65535. Il valore dei DEFAULT e' 1023.

MODO MULTICOLOR

X puo' variare da 160 a 65535. Il valore dei DEFAULT e' 511.

Y puo' variare da 160 a 65535. Il valore dei DEFAULT e' 511.

Vediamo subito due esempi applicativi

```
100 COLOR1,11
110 GRAPHIC1,1
120 SCALE1
130 CIRCLE1,160,100,50,40
```

```
100 COLOR 1,9:GRAPHIC 1,3
110 SCALE1,500,1000
120 CIRCLE 1,512,512,140,180
```

SCRATCH

FORMATO: SCRATCH "(nome)"[,Dx] [ON U (y)]

FUNZIONE: Cancella un file ad disco

AZIONE: "ARE YOU SURE?" e' il messaggio che appare sul video ed al quale l' utente dovra' rispondere con un "YES" o semplicemente "Y" e RETURN, per confermare.
In caso di mancanza di assegnazione di parametri verranno automaticamente assegnati come periferica il disco (8) e come drive lo 0 (ZERO).

ESEMPIO:

```
SCRATCH"PIPPO"
ARE YOU SURE?
YES
01,FILES SCRATCHED,01,00
READY.
```

SCNCLR

FORMATO: SCNCLR modo

FUNZIONE: Cancella lo schermo corrente

AZIONE: Questo comando serve per cancellare l' attuale visualizzazione di schermo in qualsiasi modo si desideri. Infatti e' sufficiente dare un valore al parametro MODO.

NUMERO MODO	TIPO SCHERMO
0	Testo 40 colonne
1	Bit Map MODE
2	Split Screen in Bit Map
3	Multicolor
4	Split Screen in Multicolor
5	Testo a 80 colonne

Il comando senza parametro esegue il CLEAR sullo schermo grafico.

ESEMPI

SCNCLR 1 Esegue la pulizia dell' Alta Risoluzione

SCNCLR 5 Esegue la pulizia di schermo in modo 80 colonne.

SLEEP

FORMATO: SLEEP N

FUNZIONE: Genera ritardi

AZIONE: Serve per generare un ritardo

programmabile nell' esecuzione di un programma. Il parametro N che puo' assumere valori compresi nell' intervallo fra 0 e 65535 sta ad indicare i secondi.

ESEMPIO

```
.
.
190 SLEEP 600
.
```

SLOW

FORMATO: SLOW

FUNZIONE: Riporta il CBM 128 a 1 MHz

AZIONE: Il CBM 128 ha uno dei suoi microprocessori, il 8502 capace di girare a due velocita', cioe' a 1 o 2 Mhz.
Con questo comando si fa girare il processore a 1 Mhz, cioe' alla velocita' piu' bassa. Con il comando FAST invece si fa girare il processore piu' veloce a 2 Mhz. Contrariamente a quanto affermato cio' non ha nessun effetto sulle periferiche.

SOUND

FORMATO: SOUND v,f,d[,dir][,m][,s][,w][,p]

FUNZIONE: Controlla le uscite del suono e le note musicali.

AZIONE: Vediamo innanzi tutto i parametri ed i valori ad essi assegnabili:

v = voce (1,2,3)

f = controllo frequenza (da 0 a 65535)

d = durata (da 0 a 32767)

dir = direzione dove 0= UP, 1= DOWN, 2= OSCILLATE. Il valore di default sara' =0
m = frequenza minima (da 0 a 65535). Anche in questo caso il valore di default = 0.

s = valore del passo per lo SWEEP (da 0 a 32767) default = 0

w = forma d' onda in cui con 0= triangolo, 1= dente di sega, 2= variabile, 3= rumore bianco. Per default avremo il valore di 2.

p = ampiezza d' onda (da 0 a 4095). Default = 2048

Per prima cosa bisogna ricordare che la durata del suono e' misurata in JIFFIES che equivalgono ad un sessantesimo di secondo. (60 JIFFIES = 1 Secondo).
Questo comando consente di produrre suoni con una facilita' ed una velocita' notevole. Basta solo fare un po' di prove e quindi prenderci un po' di pratica.
Dei parametri appena visti V,F e D selezionano la voce, la frequenza e la durata del suono stesso. Questo parametro e' in sessantesimo di secondo.
Vediamo un po' di esempi che l' utente potra' provare e attraverso l' assegnazione di valori diversi ai parametri, notarne le differenze.

ESEMPI:

SOUND 1, 900, 720

Si produce una nota con la voce 1, con la frequenza impostata a 900 e la durata di 2 minuti.

SOUND 2, 65535, 120

Si produce una nota musicale nella voce 1 alla massima frequenza consentita per la durata di 2 secondi.

SOUND 1,6000,90,2,1000,100,0

Si producono una serie di suoni che iniziano con una frequenza minima di 1000, vanno fino ad una frequenza di 6000 con un incremento di 100. La forma d' onda selezionata e' triangolo e la voce e' la 1.

SPRCOLOR

FORMATO: SPRCOLOR [smcr -1][,smcr-2]

FUNZIONE: Fissa i colori Multicolor 1 e/o 2 per tutti gli Sprites.

AZIONE: Come al solito vediamo per prima cosa i parametri:

smcr-1 = Fissa il Multicolor 1 per tutti gli Sprites

scmr-2 = Fissa il Multicolor 2 per tutti gli Sprites

Dopo aver definito degli Sprite e quindi prima

che queste immagini possano essere utilizzate nei programmi si devono definire alcune caratteristiche fra le quali il colore.
Infatti se si impiega il modo diretto di disegno degli Sprite per creare appunto queste figure nel modo Multicolor si deve usare prima questo comando.
In caso contrario infatti l' area riservata al disegno e lo Sprite preso per campione stesso potrebbero non essere visualizzati correttamente.
Come abbiamo detto il primo parametro imposta il Multicolor 1 per tutti gli sprites ed il secondo il Multicolor 2 per tutti gli Sprites.
Ognuno di questi parametri puo' avere un codice colore da 0 a 15 corrispondenti alle tabelle gia' viste.
Se uno dei parametri di colore viene omesso il suo valore attuale di colore resta immutato.

ESEMPI

```
SPRCOLOR 1,7
```

Fissa i Multicolor 1 e 2 rispettivamente sul nero e sul bleu

```
SPRCOLOR 1,3
```

Fissa i Multicolor 1 e 2 rispettivamente sul nero e sul rosso.

```
100 M1=3:M2=7:SPRCOL M1,M2
```

Il multicolor 1 e' sul rosso il 2 sul bleu

SPRDEF

FORMATO:. SPRDEF

FUNZIONE: Imposta le caratteristiche degli sprites.

AZIONE: Con questo comando si visualizza un' area di lavoro sullo schermo di 24 x 21 che e' la massima area o griglia per ogni Sprite.
Ogni carattere che verra' posizionato su questa griglia dello schermo corrisponde ad un pixel dello sprite. In altre parole avremo una tavola su cui disegnare.
Ora ai singoli tasti saranno assegnate delle funzioni particolari che spiegheremo:

TASTI DA 1 A 8 = Selezionano lo sprite

A = Attiva o disattiva il movimento automatico del cursore

CRSR = Come al solito le freccette, a solo o con lo SHIFT, controllano i movimenti del cursore.

RETURN = Muove il cursore all' inizio della prossima linea.

HOME = Porta il cursore in alto a sinistra della griglia di lavoro

CLR = Cancella l' intera griglia

TASTI DA 1 A 4 = Servono per selezionare il colore sorgente

CRL E TASTI DA 1 A 8 = Si possono selezionare i colori da 1 a 8 per gli Sprites

COMMODORE DA 1 A 8 = Come sopra per i colori da

9 a 16

STOP = Con questo tasto si cancellano i cambiamenti fatti e si ritorna in INPUT

SHIFT RETURN = Esegue il salvataggio dello Sprite quindi torna a chiedere il Numero.

X = Consente l' espansione dello Sprite in direzione orizzontale

Y = Consente l' espansione dello Sprite in direzione verticale

M = Multicolore per lo Sprite

C = Serve per copiare i valori di uno Sprite sull' area di lavoro di un' altro Sprite.

Per gli esempi si rimanda alla sezione sulla grafica di questo manuale.

SPRITE

FORMATO: SPRITE (numero) (,on/off) (,fgnd) (,priority) (,x-exp) (,y-exp) (,mode)

FUNZIONE: Serve per controllare molte delle caratteristiche degli Sprite.

AZIONE: Vediamo subito i parametri specificati nel comando ricordando che i parametri non dichiarati assumono i valori che il precedente, stesso comando, ha assegnato. Per un controllo dei valori di questi parametri usare la Funzione RSPRITE e vedere che cosa restitituisce.

NUMERO = Attivazione del numero (1-8)

ON/OFF = Attiva (1) o disattiva (0)

FGND = Scelta colore (1-16)

PRIORITY = Possiamo scegliere fra priorita' 1 o 0. Questa priorita' e' in funzione di un oggetto sullo schermo rispetto al quale lo Sprite oggetto del nostro lavoro dovra' comportarsi in un modo o nell' altro. Se assegneremo il valore ZERO allora lo Sprite apparira' DAVANTI all7 oggetto sullo schermo. Se invece assegneremo priorita' UNO allora passera' dietro l' oggetto sullo schermo.

X-EXP = Scelta dell' attivazione (1) o disattivazione (0) dell' espansione orizzontale

Y-EXP = Scelta dell' attivazione (1) o disattivazione (0) dell' espansione verticale

MODE = Con ZERO si sceglie lo Sprite Standard con ZERO il Multicolor.

Vediamo di chiarire alcuni punti prima di passare agli esempi.
Il parametro PRIORITA' fornisce allo Sprite una priorita' di visualizzazione che lo fa APPARIRE come se si muovesse avanti o dietro la visualizzazione dello schermo quando si anima lo sprite stesso. Possiamo affermare che se il valore di priorita' e' 1 avremo una PRIORITA' BASSA e quindi lo sprite apparira' DIETRO. Peraltro possiamo definire invece come PRIORITA' ALTA se sceglieremo per questo parametro il valore 0 per cui lo Sprite apparira' DAVANTI.
In questo modo avremo la possibilita' di creare effetti tridimensionali per animare una grafica a colori.

Inoltre la relazione di PRIORITA' fra i singoli sprites e' impostata in modo tale che gli Sprites con il numero piu' basso vengano visualizzati davanti a quelli con il numero piu' alto. In altre aprole lo Sprite 1 ha la priorita' piu' alta e lo Sprite 8 quella piu' bassa.
I parametri di espansione permettono di espandere in maniera del tutto indipendente uno Sprite sino al doppio della sua dimensione in una o entrambe le direzioni o di ridurre lo Sprite alla dimensione normale.
Quando si espande uno Sprite, il numero di punti che definisce l' immagine dello Sprite non cambia, mentre invece la dimensione di ciascun punto viene raddoppiata nella direzione scelta.

ESEMPI:

SPRITE 6,1,1,0

Attiva lo Sprite 6, lo colora di nero e assegna la priorita' 0

SPRITE 1,1,3,0,1,1,1

Attiva lo Sprite 1, lo colora in Rosso. Inoltre il primo ZERO comunica al sistema di visualizzare lo Sprite davanti all' oggetto eventualmente presente sullo schermo. I due valori UNO seguenti comunicano di espandere lo Sprite sia in direzione verticale che orizzontale. L' ultimo valore 1 dichiara il Modo Multicolor. Ricordiamo che deve essere usato il comando SPRCOLOR per selezionare il Multicolor dello Sprite.

SPRSAV

FORMATO: SPRSAV (origine),(destinazione)

FUNZIONE: Sposta dati relativi ad uno Sprite

AZIONE: Questo comando serve per muovere dei dati relativi ad uno Sprite.
I tipi di operazioni che si possono effettuare con questo comando sono 2. Si possono cioe' trasferire dati da una variabile stringa ad un' area di immagazzinamento Sprite. Inoltre si possono trasferire i valori di quest' area in una variabile stringa. Per quanto riguarda i parametri sia l' orgine che la destinazione possono essere un numero di Sprite o una variabile stringa, ma non entrambi delle variabili stringa.
Ricordiamo che l' area di lavoro per uno Sprite e' pari a 63 Bytes per cui se la stringa da trasferire e' di dimensioni superiori saranno uasti solo i primi 63 Bytes.
Vediamo ora degli esempi.

ESEMPI

SPRSAV 1,2

Trasferisce i dati dall'area di immagazzinamento dello Sprite 1 a quella dello Sprite 2

SPRSAV C$,3

Trasferisce i dati contenuti nella variabile C$ comunque riempita nell' area di immagazzinamento dello Sprite 3

SSHAPE/GSHAPE

FORMATO: SSHAPE variabile stringa, X1,Y1 (X2,Y2)

GSHAPE variabile stringa (X,Y) (,modo)

FUNZIONE: Salvano e richiamano disegni impiegando delle variabili.

AZIONE: Vediamo separatamente il comando di salvataggio da quello di ricerca.
Nel comando di salvataggio SSHAPE i parametri sono i seguenti:

VARIABILE STRINGA e' il nome legale della stringa dove si devono salvare i dati
X1,Y1 sono le coordinate dell' angolo dell' area da salvare (da 0,0 a 319,199)
X2,Y2 sono le coordinate dell' angolo opposto a quello precedente. Il valore di DEFAULT e' dato dal PC.

Poiche' il Basic non puo' manipolare stringhe di dimensione maggiore di 255 caratteri la misura della stringa che si vuole salvare e di conseguenza dell' area relativa e' per forza limitata.
Per calcolare la dimensione della stringa e vedere quindi come ci si deve comportare di conseguenza si possono utilizzare le seguenti formule:

L(mcm) = INT ((ABS (a1-a2)+1)/4+99) * (ABS(b1-b2) + 4

L(h-r) = INT ((ABS (a1-a2)+1)/8+99) * (ABS(b1-b2)+1) + 4

In cui L sara' la lunghezza della stringa, (mcm) si riferira' alla modalita' multicolor e (h-r) a

quella di alta risoluzione.
La forma viene salvata riga per riga. Gli ultimi quattro bytes della stringa contengono le lunghezze della colonna e della riga meno l' unita'.

Il comando per ritrovare i dati da una variabile stringa e e visualizzarli su una data area di schermo e' GSHAPE i cui parametri, simili al precedente, indicano invece la stringa che si vuole rintracciare, le coordinate superiori sinistre dello schermo (cioe' relativamente a quell' area su cui visualizzeremo il disegno).
Il valore di DEFAULT e' il PC.
Il parametro in diverso che c' e' su questo comando e' il MODO che puo' assumere i seguenti valori con i relativi significati:

0 = Posiziona la forma cosi' come e' (e' anche il valore di DEFAULT)

1 = Visualizza il disegno in forma inversa.

2 = Esegue un OR logico fra la forma e l'area

3 = Esegue un AND logico fra la forma e l' area

4 = Esegue un XOR logico fra la forma e l' area.

Per chiarire vediamo alcuni esempi.

ESEMPI

SSHAPE "PIPPO",0,0

Salva la forma presente sullo schermo a partire dalle coordinate di valore 0,0, cioe' dal punto piu' alto a sinistra dello schermo fino al punto del PC in una variabile stringa a cui abbiamo

assegnato il nome pippo.

SSHAPE C$,0,0,100,100

Salva l' area di schermo compresa fra le coordinate di inizio e quelle di fine assegnate e immette i valori nella variabile stringa C$.

SSHAPE A$,20,20,+50,+70

Salva l' area di schermo che parte dalle coordinate 20 20 fino ai valori dati dal PC + 50 e 70.

GSHAPE "PLUTO",0,0,1

Visualizza un disegno che e' stato in precedenza memorizzato in una variabile stringa di nome PLUTO e lo pone all' inverso nell' area di schermo che va dall' inizio stesso dello schermo.

STASH

FORMATO: STASH #byte,intsa,expb,expsa

FUNZIONE: Sposta il contenuto dalla memoria principale alla espansione RAM.

AZIONE: La spiegazione e l' utilizzo di questo comando, come del resto il comando FETCH descritto in precedenza, va visto in rapporto al complesso lavoro di gestione della memoria.
I parametri sono:

Bytes = Numero dei Bytes dell' espansione di memoria. Valore accettato da 1 a 65535.

Intsa = Indirizzo di memoria della RAM interna.

Expsa = Indirizzo di memoria della memoria RAM esterna.

Expb = Numero del banco (0-3) dell' espansione RAM.

SWAP

FORMATO: SWAP #bytes,intsa,expb,expsa

FUNZIONE: Scambia i contenuti della memoria RAM interna con quelli dell' espansione esterna.

AZIONE: Per i significati dei parametri vedi il precedente comando STASH.

STOP

FORMATO: STOP

FUNZIONE: Termina l' esecuzione di un programma e ritorna nel modo comando.

AZIONE: Il comando STOP puo' essere usato in un qualsiasi punto di un programma per arrestarne l' esecuzione.
Normalmente e' utilizzato in fase di DEBUG cioe' di prova.
Quando il comando STOP viene incontrato allora sara' visualizzato il seguente messaggio:

```
BREAK IN LINE nnnn
```

Dove nnnn e' il numero di linea del programma che contiene l' istruzione STOP.
E' necessario ricordarsi pero' che il comando STOP non esegue automaticamente la chiusura dei Files.
Il computer dopo l' esecuzione di questo comando ritorna sempre nel modo comando.
L' escuzione di un programma puo' riprendere con il comando CONT (vedi).

ESEMPIO:

```
10 INPUT A,B,C
20 K=(A+3)/2:L=B*3
30 STOP
40 M=C*K+100:PRINT M

RUN

? 1,2,3        Dati inseriti
BREAK IN 30    Fermata
READY

PRINT L        Richiesta valore di L
6
READY

CONT           Ordine di continuare.

106

READY
```

SYS

FORMATO: SYS (nome della variabile) [(elenco degli argomenti)]

FUNZIONE: Per chiamare una routine in linguaggio macchina. (A questo proposito vedi anche la funzione USR).

AZIONE: Il nome della variabile contiene un' indirizzo che e' il punto di partenza della routine memorizzata.
La lista degli argomenti contiene la lista degli argomenti che possono essere passati ad una subroutine in Linguaggio Macchina.
La lista degli argomenti, contenuti nella lista opzionale puo' essere dei seguenti significati:

Prima posizione = il valore che deve essere caricato nell' ACCUMULATORE.

Seconda posizione = il valore che deve essere caricato nel REGISTRO X

Terza posizione = il valore che deve essere caricato nel REGISTRO Y.

Quarta posizione = il valore che deve essere caricato nello STATUS REGISTER.

Per i significati di queste parole vedi la successiva sezione relativa al linguaggio macchina.

ESEMPIO:

```
110 MYROUT=30200
120 SYS MYROUT (A1)
```

Quando il programma arriva alla linea 120 l' esecuzione passera' dal Basic alla routine in Linguaggio Macchina di indirizzo decimale 30200.

SYS 8096,0,1,0,1

Si andra' ad eseguire la routine di indirizzo 8096 e verranno caricati nell' Accumulatore il valore 0, in X 1, in Y 0 e nello Status register 1.

TEMPO

FORMATO: TEMPO n

FUNZIONE: Definisce la velocita' dell' uscita sonora.

AZIONE: Da il tempo al suono. Infatti, uno dei parametri relativi al suono e' dato appunto dalla durata. Il parametro n che puo' assumere valori fra 0 e 255 determina la durata relativa della nota.
Si puo' determinare la durata dell' intera nota utilizzando la seguente formula:

DURATA DELL' INTERA NOTA = 19.22/n secondi

Il valore di DEFAULT e' 8 mentre la durata della nota si incrementa con n.

ESEMPI

TEMPO 40 Definisce un tempo di 40

TEMPO 255 Definisce la durata massima

TEMPO 1 Definisce la durata minima

TRAP

FORMATO: TRAP (linea)

FUNZIONE: Intercetta le condizioni di errore

AZIONE: Quando viene attivato TRAP consente di intercettare tutte le condizioni di errore incluso il tasto di RUN STOP, ma escluso gli errori DOS ed escluso il " UNDEF'D STATEMENT ERROR".
Nel caso di un qualsiasi errore di esecuzione, eccetto quelli visti, viene visualizzato il flag d' errore e l' esecuzione del programma passa al numero di riga definito come parametro dell' istruzione TRAP.
Si puo' individuare il numero della riga in cui si e' avuto l' errore utilizzando la variabile di sistema EL, mentre la condizione di errore specifica e' contenuta nella variabile di sistema ER.
La funzione stringa ERR$(ER) fornisce il messaggio di errore corrispondente ad una qualsiasicondizione di errore.
Da notare in particolare che il comando RESUME puo' essere utilizzato per far riprendere l' esecuzione del programma, mentre un errore contenuto in una routine di TRAP non puo' essere individuato.
Quando si usa TRAP senza il numero di linea si disattiva il cosi' detto sistema di TRAPPOLA DI ERRORE.

ESEMPIO

```
10  TRAP 100
.
.
100 PRINT ERR$(ER);EL
110 RESUME
```

In questo caso la linea 10 consente di saltare alla subrotine di indirizzo 100 per la visualizzazione rispettivamente del messaggio di errore e del numero. La linea 110 fa ripartire il programma dall' istruzione immediatamente successiva a quella in cui si e' verificato l' errore.

TROFF

FORMATO: TROFF

FUNZIONE: Disattiva il modo TRACE

AZIONE: Per maggiori dettagli vedi l' istruzione seguente.

TRON

FORMATO: TRON

FUNZIONE: Attiva il modo TRACE

AZIONE: Con questa istruzione si attiva il modo TRACE che e' impiegato normalmente nella fase di sviluppo programmi.
Quando con questa istruzione il modo TRACE e' attivato, mano a mano che le istruzioni di un programma vengono eseguite, viene visualizzata il numero di riga relativo a quell' istruzione fra parentesi quadre.

VERIFY

FORMATO: VERIFY ["(nome del file)" [,(periferica)]]

FUNZIONE: Confronta il contenuto di un programma in memoria con quello di un file su disco o nastro e ne riporta le differenze.

AZIONE: In mancanza di specificazione circa il numero della periferica la ricerca e il controllo verranno effettuati sulla cassetta n.1.

ESEMPIO:

VERIFY"PIPPO"

PRESS PLAY ON TAPE 1

OK

VERIFYING

VERIFY ERROR

READY.

E' stato trovato un errore di tipo qualsiasi.

VOL

FORMATO: VOL livello

FUNZIONE: Definisce il livello di uscita sonoro.

AZIONE: Questo comando fissa il volume in

relazione all' impiego dei comandi SOUND e PLAY. VOL agisce su tutte le voci attive.
Il parametro livello puo' essere fissato fra i valori 0 e 15 dove 15 e' il massimo volume possibile, 1 il minimo e 0 e' invece disattivato.

ESEMPI

VOL 1 Fissa il volume al minimo

VOL 0 Cancella il volume

VOL 10 Fissa il volume a 10

Anche in questo caso la cosa migliore e' fare delle prove.

WAIT

FORMATO: WAIT Indirizzo,I [,J]

FUNZIONE: Sospende temporaneamente l' esecuzione di un programma mentre sta controllando lo stato della porta di ingresso del computer.

AZIONE: Il comando WAIT consente la sospensione dell' esecuzione di un programma fino a quando un dato indirizzo in codice macchina non sviluppi, o meglio non abbia come risultato, un dato valore.
Sul dato letto a quell' indirizzo viene eseguito un OR esclusivo con l' espressione intera J e poi un AND con I.
Se il risultato di questa operazione e' ZERO, il Basic torna indietro e legge ancora i dati dalla porta.
Se il risultato e' diverso da ZERO allora l'

esecuzione del programma continua con il comando successivo.
Nel caso sia omesso il valore di J nel comando questi viene assunto come ZERO.

ATTENZIONE

E' possibile con questo comando entrare in un LOOP, cioe' in un ciclo infinito. In questo caso l' unica cosa da fare e' di eseguire manualmente il RESTART del computer.

ESEMPIO:

```
100 WAIT 1,32,32
```

Questo comando sospende l' esecuzione di un programma fino a quando non sia premuto il tasto PLAY sull' unita' a cassette.

WIDTH

FORMATO: WIDTH n

FUNZIONE: Fissa l' ampiezza delle linee

AZIONE: Quando si utilizzano i comandi grafici del Basic del 128 si puo' scegliere se disegnare con linee fini o grosse. I valori assegnabili al parametro di questo comando sono solo due. Se si assegna il valore 1 avremo una linea di dimensioni normali, con il valore 2 avremo una linea doppia della precedente.

WINDOW

FORMATO: WINDOW X1,Y1,X2,Y2 (clear)

FUNZIONE: Definisce delle finestre di schermo

AZIONE: Questo comando, che puo' essere usato in forma multipla, puo' definire finestre sullo schermo sia quando operiamo su 40 che su 80 colonne.
I parametri X che possono assumere 0/39 o 0/79 a secondo se operiamo su 40 o 80 colonne definiscono le coordinate delle colonne, mentre gli Y definiscono quelle delle righe e potranno assumere valori da 0 a 24.
Se diamo 1 al parametro CLEAR avremo una funzione di CLEAR di schermo nell' ambito della finestra.

ESEMPI

WINDOW 0,0,25,10

Definisce una finestra con coordinate dell' angolo a sinistra 0,0 e 25,10 a destra in basso sullo schermo.

WINDOW 5,10,35,20,1

Definisce una finestra di schermo come sopra, ma prima di visualizzarci i dati ne esegue anche la pulizia (cioe' un CLEAR).

FUNZIONI BASIC DEL C128

In questa sezione sono presentate le funzioni intrinseche del Basic 7.0 implementato sul C128.

L' argomento di queste funzioni e' sempre racchiuso fra parentesi.
Nel formato dato per le funzioni in questa sezione, gli argomenti sono stati abbreviati come segue:

X e Y Rappresentano una stringa numerica qualsiasi.

I e J Rappresentano un' espressione intera.

X$ e Y$ Rappresentano un' espressione stringa.

Se viene fornito un valore in virgola mobile quando invece e' richiesto un' intero, il Basic tronchera' la parte frazionaria ed utilizzera' il risultato intero.

ABS

FORMATO: ABS(X)

FUNZIONE: Riporta il valore assoluto dell' espressione X

AZIONE: La funzione calcolera' il valore dell' espressione contenuta fra parentesi come un

numero positivo e non vi sara' nessuna perdita di precisione.

ESEMPIO:

```
PRINT ABS(7*(-5))

    35

READY
```

ASC

FORMATO: ASC(X$)

FUNZIONE: Riporta il valore numerico che e' il codice ASCII del primo carattere della stringa **X$**.
Se X$ ha un valore nullo, viene visualizzato un errore:

ILLEGAL QUANTITY

ESEMPIO:

```
10 X$="TEST"
20 PRINT ASC(X$)

RUN

      84

READY
```

NOTA

Nell' esempio precedente 84 e' il valore ASCII di T che e' il primo carattere della stringa.

Vedi anche la funzione CHR$ per la conversione.

ATN

FORMATO: ATN(X)

FUNZIONE: Riporta l' arcotangente di X. Il risultato sara' in radianti in un' intervallo fra -pigreco/2 e +pigreco/2.
L' espressione X puo' essere un qualsiasi numero, ma lo sviluppo e quindi la valutazione di ATN e' sempre eseguita in virgola mobile binaria.

ESEMPIO:

```
10 INPUT X
20 PRINT ATN(X)

RUN

? 3

1.24904577

READY
```

BUMP

FORMATO: BUMP(n)

FUNZIONE: Restituisce informazioni sulla collisione fra Sprites.
Viene impiegata questa funzione per determinare quale o quali Sprite hanno avuto una collisione.
Se assegnamo a N il valore 1 avremo un rapporto

sulla collisione fra 2 sprite. Se invece si assegna ad N il valore 2 potremo conoscere invece quale sprite ha avuto una collisione con un altro oggetto qualsiasi sullo schermo.
Ricordiamoche non e' necessario attivare il comando COLLISION per avere risultati nell' uso di questa funzione e BUMP viene riportato a zero dopo ogni chiamata in uso.
Vediamo, con l' aiuto della seguente tabella di comprendere i valori restituito dalla funzione.

SPRITE	1	2	3	4	5	6	7	8
VALORE	1	2	4	8	16	32	64	128

Ad ogni Sprite e' quindi assegnato un valore. Vediamo gli esempi.

ESEMPI

```
PRINT BUMP (1)

     48
```

Avremo quindi che due Sprites sono entrati in collisione. Saranno quindi gli Sprites 5 e 6 la somma dei cui valori fda appunto 48 (16+32).

```
PRINT BUMP (1)

     6
```

Gli Sprites entrati in collisione sono il 2 ed il 3 (2+4)

```
PRINT BUMP (2)

     64
```

In questo caso lo Sprite 7 ha avuto una

collisione con un altro oggetto sullo schermo.

CHR$

FORMATO: CHR$(I)

FUNZIONE: Riporta una stringa di cui un elemento ha il codice ASCII I.
CHR$ e' comunemente usata per inviare un carattere speciale ad una periferica. Ad esempio con CHR$(147) si esegue la pulizia di schermo (il CLEAR SCREEN) ed il cursore viene riportato nella HOME POSITION che e' la posizione piu' in alto a sinistra dello schermo.

ESEMPIO:

```
PRINT CHR$(66)

B

READY.
```

NOTA

Per i valori assegnabili a CHR$ ed i significati vedi l' apposita tavola al termine di questo manuale.

COS

FORMATO: COS(X)

FUNZIONE: Riporta il coseno di X in radianti.
Il calcolo di COS(X) e' eseguito in virgola mobile binaria.

ESEMPIO:

```
10 X=2*COS(.4)
20 PRINT (X)

RUN

1.84212199

READY.
```

DEC

FORMATO: DEC(stringa esa)

FUNZIONE: Restituisce il valore decimale di un numero in esadecimale espresso come stringa.

ESEMPIO

```
PRINT DEC("D020")

53280
```

Ricordiamo di scriverlo correttamente perche' altrimenti avremo un errore.

DS

FORMATO: DS

FUNZIONE: Riporta il codice di errore corrispondente all' ultima operazione su disco.

Si consiglia di consultare la sezione relativa

alle periferiche e la parte di questo volume riguardante gli errori.

Se nessuna periferica risponde al comando DS vuol dire che il valoore di DS sara' nell' intervallo fra 0 e 20.

ESEMPIO:

```
20 IF DS>=20 THEN PRINT "ERROR": STOP
```

DS$

FORMATO: DS$

FUNZIONE: Riporta la stringa dello stato del disco (appunto la Disk Status String) corrispondente all' ultima operazione eseguita sul disco dal computer.
La stringa sara' composta dal codice di errore, dal messaggio, dalla traccia e dal settore. I campi, come si puo' vedere dall' esempio, sono separati da virgola.

Se la periferica disco non e' presente allora LEN(DS$)=0.

ESEMPIO:

```
PRINT DS$

00,OK,00,00
```

Anche in questo caso si consiglia di consultare sia il manuale relativo all' uso del disco sia la parte delle periferiche su questa Guida.

ERR$

FORMATO: ERR$(N)

FUNZIONE: Restituisce una stringa che descrive una condizione di errore. Per la lista completa ed il significato ed i possibili rimedi degli errori basic vedi l' appendice dedicata a questo scopo.

ESEMPIO

```
PRINT ERR$(ER)

syntax error
```

Notare che questo potra' essere uno dei tanti errori.

EXP

FORMATO: EXP(X)

FUNZIONE: Riporta e alla potenza di X dove X deve essere minore o uguale a 88.02969191. Se X e' maggiore verra' visualizzato un messaggio di errore:

"OVERFLOW"

ESEMPIO:

```
10 X=5
20 PRINT EXP(X-1)

RUN

54.5981501
```

READY.

FRE

FORMATO: FRE(X)

FUNZIONE: Il parametro o argomento di FRE indica il numero del Banco di memoria su cui si esegue il controllo.
Quindi la funzione FRE(0) riporta il numero di Bytes in memoria che non sono stati usati dal Basic. FRE(1) indica il numero di Bytes liberi nell' area di immagazzinamento variabili.

ESEMPIO:

```
PRINT FRE(0)
14542
READY
```

ATTENZIONE

Il risultato puo' essere un numero negativo nel CBM64. Consultare attentamente il manuale in proposito.

HEX$

FORMATO: HEX$(X)

FUNZIONE: Restituisce un numero esadecimale sotto forma di stringa da un numero decimale. Il valore da dare ad X deve essere compreso fra 0 e 65535.

ESEMPIO

```
PRINT HEX$(53280)

    D020
```

Da notare che per effettuare la conversione opposta impiegheremo il gia' visto comando DEC.

INSTR

FORMATO: INSTR(stringa1,stringa2[,posizione di partenza])

FUNZIONE:Questa utilissima funzione effettuala ricerca del contenuto della stringa 2 all' interno della stringa 1.
Il parametro opzionale POSIZIONE DI PARTENZA stabilisce da dove iniziare la ricerca e puo' essere un valore compreso nell' intervallo fra 1 e 255.
Se la ricerca ha esito negativo, oppure se la posizione di partenza e' piu' grande della stringa 1 oppure se la stringa 1 e' nulla avremo INSTR=0.

ESEMPIO

```
PRINT INSTR("VIA MARCONI 9/A","9/A")

      13
```

Che corrisponde alla posizione iniziale della seconda stringa.
Notare che se la stringa 2 e' nulla allora INSTR restituisce il valore della posizione di partenza.

INT

FORMATO: INT(X)

FUNZIONE: Restituisce il valore intero di un'espressione X. Se l' espressione e' positiva allora la parte frazionaria viene tagliata fuori. Se l' espressione e' negativa avremo come ritorno il numero intero minore piu' vicino.

ESEMPI

```
PRINT INT(99.89)

99

READY.

PRINT INT(-12.11)

-13

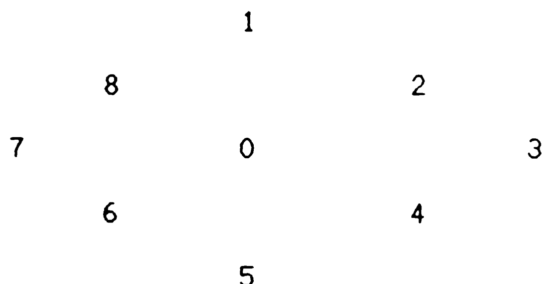
READY.
```

JOY

FORMATO: JOY(N)

FUNZIONE: Permette di leggere la posizione di una o due joystick.
Il parametro N infatti puo' assumere il valore 1 o 2 e specifica quindi quale delle joystick in lettura e' connessa alla porta di controllo 1 o 2. Il numero che e' restituito da questa fase di lettura indica verso quale direzione ci si muove. Quando viene premuto il tasto FIRE (fuoco), verra' aggiunto 128 al valore indicatore della direzione.

Lo schema seguente mostra i valori che corrispondono alle varie direzioni:



ESEMPIO

```
10 H=JOY(1):IF H=128 THEN GOSUB1000
```

L' istruzione significa che se il tasto FUOCO del joystick connesso alla porta 1 e' stato premuto, allora il programma andra' ad eseguire la routine della linea 1000

JOY (2) = 131

Il joystick collegato alla porta 2 e' spostato verso destra ed il bottone di fuoco e' premuto (128+3 = 131)

LEFT$

FORMATO: LEFT$(X$,1)

FUNZIONE: Riporta una stringa che comprende il carattere piu' a sinistra I di X$.
I deve essere compreso nell' intervallo fra 0 e 255.
Se I e' piu' grande di LEN(X$) allora sara' restituita l' intera stringa X$.
Se I=0 verra' riportata una stringa nulla, cioe' di lunghezza zero.

ESEMPIO:

```
10 A$ ="COMMODORE BASIC"
20 B$=LEFT$(A$,9)
30 PRINT B$

          COMMODORE
```

NOTA

Vedi a proposito del trattamento delle stringhe anche le funzioni MID$ e RIGHT$.

LEN

FORMATO: LEN(X$)

FUNZIONE: Riporta il numero di caratteri contenuti in X$.
Sono conteggiati sia caratteri non stampabili che gli spazi bianchi.

ESEMPIO:

```
10 X$="SANTA CLARA, CALIFORNIA"
20 PRINT LEN(X$)

        23
```

LOG

FORMATO: LOG(X)

FUNZIONE: Restituisce il logaritmo naturale di X.
X deve essere maggiore di zero.

ESEMPIO:

```
PRINT LOG(45/7)

1.86075234
```

NOTA

Ricordiamo che il logaritmo NATURALE e' il logaritmo in base e. Per convertire al logaritmo in base 10 occorre dividere per LOG(10)

MID$

FORMATO: MID$(X$,I[,j])

FUNZIONE: Riporta una stringa della lunghezza di J caratteri dalla stringa in esame X$ iniziando con l' Iesimo carattere.
I e J deveono essere compresi nell' intervallo fra 0 e 255.

Se J e' omesso o se ci sono meno di J caratteri alla destra dell' Iesimo carattere, allora sono riportati tutti i caratteri piu' a destra iniziando dall' Iesimo carattere.
Se I e' maggiore di LEN(X$), allora la funzione MID$ restituisce una stringa nulla.

ESEMPIO:

```
LIST

10 A$="GOOD"
20 B$="MORNING EVENING AFTERNOON"
30 PRINT A$;MID$(B$,9,7)

RUN

GOOD EVENING
```

PEEK

FORMATO: PEEK(I)

FUNZIONE: Riporta il Byte (intero decimale compreso nell' intervallo da 0 a 255) letto dalla locazione di memoria I.
I deve essere compreso nell' intervallo fra 0 e 65535.
PEEK e' la funzione complementare del comando POKE.
Normalmente ci troviamo ad operare sul banco di memoria 15 ed e' quindi su questo banco che avviene la lettura.
Se ci troviamo ad operare su altro banco dobbiamo fare attenzione perche' la lettura viene fatta appunto sul banco selezionato per ultimo. Vedi anche il comando BANK.

ESEMPI

```
PRINT PEEK(59468)
```

E riportera' un valore che potra' essere qualsiasi.

```
10 BANK 3
20 PRINT PEEK(40000)
```

In questo caso la lettura verra' fatta sul banco 3 che e' un banco di RAM Interna.

PEN

FORMATO: PEN(n)

FUNZIONE: Restituisce le coordinate X e Y della LIGHT PEN.
I valori attribuibili al parametro N sono i seguenti con i relativi significati:

0 = Riporta la coordinata X della posizione della LIGHT PEN

1 = Riporta la coordinata Y della posizione della LIGHT PEN

2 = Riporta la coordinata X della posizione della LIGHT PEN quando siamo in modo 80 colonne

3 = Riporta la coordinata Y della posizione della LIGHT PEN quando siamo in modo 80 colonne

4 = Riporta il valore di TRIGGER della LIGHT PEN

E' importante osservare per una corretta interpretazione dei valori, che le cifre fornite da PEN non sono in scala e quindi utilizzano coordinate reali e non della grafica in BIT MAP.
La posizione X riportata sara' un numero approssimativamente compreso fra 60 e 320, mentre la Y sara' fra 50 e 250. Questi infatti sono le coordinate visibili sullo schermo, mentre gli altri valori non sono visibili.
Un valore di 0 per entrambe le posizioni significa che la punta della penna luminosa non si trova sullo schermo oppure si trova a distanza tale da non avere impulsi significativi.
Chi impiega questa funzione di cui riportiamo sotto gli esempi, dovrebbe inoltre ricordare le seguenti note e particolarita'.

1) Viene richiesto uno sfondo bianco o comunque molto luminoso per attivare la lettura delle posizioni.
2) I valori possono variare da un sistema all' altro.
3) I valori possono variare da uno schermo all' altro.
4) Per quanto riguarda lo schermo ad 80 colonne i valori non sono dei PIXEL come sullo schermo a 40 colonne ma sono la posizione, sempre riga e colonna, del carattere sullo schermo.
5) Anche cambiando il tipo di LIGHT PEN e' necessario, sempre sullo stesso sistema effettuare la cosi' detta TARATURA della penna.

ESEMPIO

```
PRINT PEN(0);PEN(1)
```

Visualizza le coordinate X e Y della penna luminosa.

POINTER

FORMATO: POINTER(nome della variabile)

FUNZIONE: Riporta l' indirizzo del nome di una variabile che verra' indicata nel parametro delcomando.

ESEMPIO

```
1000 C = POINTER (H)
```

POS

FORMATO: POS(X)

FUNZIONE: Riporta l' attuale posizione del cursore. Ricordiamo che la posizione piu' a sinistra e' ZERO e che X e' un argomento DUMMY.

ESEMPIO:

```
IF POS(X)>60 THEN PRINT CHR$(13)
```

POT

FORMATO: POT (n)

FUNZIONE: Riporta il valore delle paddles. I valori ed i significati che si possono attribuire al parametro fra parentesi sono elencati nella seguente tabella:

n = 1 riporta la posizione della PADDLE #1
n = 2 riporta la posizione della PADDLE #2

n = 3 riporta la posizione della PADDLE #3
n = 4 riporta la posizione della PADDLE #4

Il valore restituito sara' un numero compreso nell' intervallo fra 0 e 255. Un qualsiasi valore maggiore di 256 indica che il pulsante di fuoco e' premuto.
Vediamo un esempio che visualizza il valore della PADDLE 3

ESEMPIO:

```
100 POT (3)
110 IF POT(3) = 256 GOTO 1000
```

RCLR

FORMATO: RCLR (n)

FUNZIONE: Questa funzione restituisce il colore, (ricordiamo che sara' un valore da 1 a 16 e guardare la tabella colori) assegnato al colore sorgente. Per questo N potra' avere i seguenti valori esignificati:

0 = Interno sul modo 40 colonne
1 = Esterno sul modo bit map
2 = Multicolore 1
3 = Multicolore 2
4 = Bordo sul 40 colonne
5 = Colore del carattere sul 40 o 80 colonne
6 = Interno sull' 80 colonne

Ricordiamo che la controparte di questa funzione e' il comando COLOR a cui si rimanda per

successivi approfondimenti.

ESEMPIO

```
 90 A=1
100 PRINT" IL SORGENTE E'";A;RCLR (A)
```

RDOT

FORMATO: RDOT (n)

FUNZIONE: Restituisce l' attuale posizione o il colore sorgente del PC. Il parametro N puo' assumere i seguenti valori:

0 = Riporta la coordinata X del PC
1 = Riporta la coordinata Y del PC
2 = Riporta il colore sorgente del PC

Il PC come abbiamo gia' detto e' il PIXEL CURSOR.

RGR

FORMATO: RGR (n)

FUNZIONE: Riporta il modo grafico in cui si sta attualmente operando. Il parametro N in questo caso e' un argomento DUMMY ma che deve essere specificato. I valori restituiti da RGR avranno i seguenti significati:

0 = Modo testo a 40 colonne

1 = Modo bit map standard
2 = Modo Bit map con Split screen
3 = Modo Multicolor bit map
4 = Modo Multicolor bit map con Spli screen
5 = Modo 80 colonne testo

La controparte di questa funzione, cioe' il comando che e' stato utilizzato per selezionare un dato modo grafico e' GRAPHIC.

ESEMPIO

```
100 PRINT RGR (0)
```

A secondo di che modo grafico stiamo adoperando ci verra' quindi restituito un valore da 0 a 5.

RIGHT$

FORMATO: RIGHT$(X$,I)

FUNZIONE: Riporta il carattere I piu' a destradella stringa X$. Se I=LEN(X$), allora restituisce l' intera stringa X$.
Se invece I=0, viene riportata una stringa nulla.

ESEMPIO:

```
10 A$="COMMODORE BASIC"
20 PRINT RIGHT$(A$,5)

RUN

        BASIC
```

RND

FORMATO: RND(X)

FUNZIONE: Riporta un numero casuale nell' intervallo fra 0 e 1.
Se X e' maggiore di 0 restituisce la stessa sequenza di numeri pseudo-casuali per ogni numero casuale selezionato.
Se X e' minore di ZERO la sequenza viene riselezionata ed ogni X produce una diversa sequenza.
La sequenza e' selezionata casualmente all' accensione del sistema.

X=0 genera un vero numero casuale prelevando i dati dal CLOCK interno.

ESEMPIO:

```
10 FORI=1TO5
20 PRINT INT(RND(X)*100);
30 NEXT

RUN

        24  30  31  51  5
```

RSPCOLOR

FORMATO: RSPCOLOR (registro)

FUNZIONE: Riporta il valore dello sprite quando questi e' in modo multicolor. Il parametro registro puo' avere i valori:

1 = Riporta lo Sprite multicolor 1

2 = Riporta lo Sprite multicolor 2

Il valore restituito sara' un numero fra 1 e 16 in rapporto al colore che stiamo usando.
La controparte della funzione e' il comando SPRCOLOR.

ESEMPIO

```
PRINT"LO SPRITE 1 E'";RSPCOLOR(1)
PRINT"LO SPRITE 2 E'";RSPCOLOR(2)
```

E quando queste righe saranno inserite in un programma si potra' vedere i valori dei colori dei due AREA SPRITE

RSPPOS

FORMATO: RSPPOS(n. sprite, posizione/velocita')

FUNZIONE: Restituisce la velocita' e i valori di posizione di uno Sprite.
Nei parametri dopo la funzione deve essere indicato il numero dello sprite che si desidera controllare, la posizione dichiara le coordinate X e Y dello Sprite. Come ultimo parametro puo' essere richiesta la velocita'. La scelta fra posizioni sulle coordinate e velocita' viene fatta attribuendo un dato valore alla seconda parte del parametro:

0 = Riporta l' attuale posizione sull' asse X di quel dato Sprite.
1 = Riporta l' attuale posizione sull' asse Y di quel dato Sprite.
2 = Riporta la velocita' da 0 a 15 sempre di quel dato Sprite.

ESEMPIO

```
100 PRINT RSPPOS(2,0);RSPPOS(2,1);RSPPOS(2,2)
```

I valori che verranno stampati in conseguenza di questo comando saranno rispettivamente le coordinate X e Y e la velocita' dello Sprite numero 2.

RSPRITE

FORMATO: RSPRITE(n. sprite, caratteristiche)

FUNZIONE: Restituisce i valori cartteristici di uno sprite, valori che erano stati impostati con il comando SPRITE (vedi).

ESEMPIO:

```
100 FOR A=0TO5
110 PRINT RSPRITE (3,A);
120 NEXT
```

Facendo girare questo piccolo programma o inserendolo come subroutine in un programma di dimensioni maggiori avremo visualizzati su una riga le caratteristiche dello Sprite numero 3.

RWINDOW

FORMATO: RWINDOW (n)

FUNZIONE: Riporta le dimensioni della finestra di schermo selezionata.
In funzione di valori che assegneremo al parametro N avremo come risposta i seguenti

risultati:

0 = Riporta il numero di linee della finestra.
1 = Riporta il numero di righe della finestra.
2 = Riporta il tipo di schermo, se cioe' a 40 o a 80 colonne.

La controparte di questo comando e' il comando WINDOW con cui si fissano i valori della finestra.

ESEMPIO

Avendo fissato una finestra di 15 per 20 su schermo a 80 colonne con:

```
PRINT RWINDOW(0);RWINDOW(1);RWINDOW(2)
```

avremo come visualizzazione i seguenti valori:

```
     15       20       80
```

SGN

FORMATO: SGN (X)

FUNZIONE: Se X e' maggiore di 0, SGN(X) riporta 1, se X=0 riporta 0, se X e minore di 0 riporta -1.

ESEMPIO:

```
ON SGN(X)+2 GOTO100,200,300
```

Cioe' salta alla linea 100 se X e' negativo, alla 200 se X e' 0, alla linea 300 se X e'

positivo.

SIN

FORMATO: SIN(X)

FUNZIONE: Riporta il seno di X in radianti. Il seno di X e' calcolato in virgola mobile binaria.
Il coseno di X sara':

```
COS(X) =SIN(X+3.14159265/2)
```

ESEMPIO:

```
PRINT SIN(1.5)

     .997494987
```

SPC

FORMATO: SPC(I)

FUNZIONE: Stampa tanti spazi bianchi (BLANKS) su di una periferica (di solito il video) iniziando dalla posizione del cursore.
Questa funzione puo' essere solo usata con il comando PRINT e deve essere nell' intervallo fra 0 e 255.

ESEMPIO:

```
PRINT"OVER"SPC(15)"THERE"

OVER               THERE
```

READY.

SQR

FORMATO: SQR(X)

FUNZIONE: Riporta la radice quadrata di X e X deve essere maggiore o uguale a zero.

ESEMPIO:

```
10 FOR X=10TO25 STEP5
20 PRINT X,SQR(X)
30 NEXT
```

RUN

```
10        3.16227766
15        3.87298335
20        4.47213595
25        5
```

STATUS

FORMATO: STATUS

FUNZIONE: Riporta lo stato del computer corrispondente all' ultima operazione di I/O, sia su cassetta, schermo,tastiera .

ESEMPIO:

```
10 DOPEN#2,"MASTER FILE"
12 GET#2,A$
14 IF STATUS AND 64 THEN 20
16 PRINTA$
```

```
18 GOTO 20
20 PRINTA$:DCLOSE#2
```

STR$

FORMATO: STR$(X)

FUNZIONE: Restituisce la rappresentazione stringa del valore di X.

ESEMPIO:

```
10 X=1024.25
20 PRINT STR$ (X)
```

RUN

Verra' visualizzato : 1024.25 Preceduto da uno spazio bianco.

TAB

FORMATO: TAB(I)

FUNZIONE: Salta alla posizione I sulla periferica.
Se l'attuale posizione di stampa e' gia' oltre lo spazio I, allora TAB fa andare a quella stessa posizione ma sulla linea successiva.
Lo spazio 0 e' la posizione piu' a sinistra mentre la posizione piu' a destra e' data dalla larghezza meno 1.
I deve essere un valore compreso nell' intervallo fra 0 e 255.
La funzione TAB puo' essere usata solo con il comando PRINT.

ESEMPIO:

```
10 PRINT"NOME"TAB(25)"AMMONTARE":PRINT
20 READ A$,B$
30 PRINTA$,TAB(25)B$
40 DATA"G. ROSSI","L.50.000"
```

```
RUN
NOME                AMMONTARE


G. ROSSI            L.50.000
```

TAN

FORMATO: TAN(X)

FUNZIONE: **Riporta la tangente di X in radianti. TAN(X) e' calcolata in binario.**

ESEMPIO:

```
10 X=.785398163
20 Y=TAN(X)
30 PRINT Y
```

```
RUN

            1
```

TIME

FORMATO: TI

FUNZIONE: E' utilizzato **per leggere** il **CLOCK**

(orologio) interno e riporta il valore in sessantesimi di secondo.

ESEMPIO:

```
10 ON TI GOTO 100,200,300
```

TIMES

FORMATO: TI$

FUNZIONE: E' utilizzato per leggere il CLOCK interno e riporta una stringa di 6 caratteri con i primi due caratteri che sono le ore, i due successivi i minuti e gli ultimi due i secondi. Puo' essere usato in un comando di input oppure sulla parte sinistra di un' espressione per fissare il tempo.

ESEMPIO:

```
10 TI$"000000"
20 FORI=1TO10000:NEXT
30 PRINT TI$
```

RUN

000011

USR

FORMATO: USR(X)

FUNZIONE: Serve per chiamare una subroutine in linguaggio macchina il cui argomento e' X. Ricordiamo che quando viene usata questa

funzione, si salta ad un programma in linguaggio macchina il cui punto di inizio e' agli indirizzi di memoria decimali 4633 e 4634.
Il valore del parametro X viene passato all' Accumulatore in virgola mobile.

ESEMPIO:

```
40 B=T*SIN(Y)
50 C=USR(B/2)
60 D=USR(B/3)
```

VAL

FORMATO: VAL(X$)

FUNZIONE: Riporta il valore numerico della stringa X$.
Se il primo carattere della stringa X$ non e' +,-,$ o un digit allora VAL (X$)=0
E' la funzione complementare di STR$ e converte una stringa in un numero che puo' essere utilizzato per i calcoli.
Se il primo carattere diverso dallo spazio che si incontra nella stringa in esame e' un carattere non numerico, allora la funzione VAL restituira' un valore nullo.
In altre parole la funzione VAL puo' convertire tanti caratteri numerici quanti ne incontra prima di trovare un carattere non valido.

ESEMPIO:

```
PRINT VAL ("3.1415 CBM")
```

sara' visualizzato:

```
3.1415
```

XOR

FORMATO: XOR(n1,n2)

FUNZIONE: Riporta un OR esclusivo fra i valori assegnati a N1 e N2.

ESEMPIO

```
PRINT XOR(128,64)
```

Verra' visualizzato il numero 192.

CAPITOLO SECONDO

IL BASIC

In questo capitolo tratteremo del BASIC in generale e di come manipoli alcune funzioni essenziali alla programmazione.
Il Sistema Operativo di questo computer ha essenzialmente le seguenti funzioni:

1) Consentire il funzionamento dell' insieme dei circuiti interagenti.

2) Consentirne il colloquio con il mondo esterno.

Il Sistema Operativo fornisce 2 modi di operare con il linguaggio Basic:

1-MODO DIRETTO

2-MODO PROGRAMMA

Usando il modo diretto le istruzioni basic non hanno numero di linea che le precede e vengono eseguite non appena viene premuto il tasto di RETURN che serve per far passare l'informazione, generalmente considerata, dalla memoria video del computer ai registri interessati.
Il modo programma e' quello usato per far girare i programmi che sono stati scritti o che vengono caricati da una peririferica.
Impiegando questo modo tutte le istruzioni devono avere all'inizio un numero che e' chiamato appunto numero di linea. Si possono

avere piu' istruzioni Basic su una stessa linea purche' siano separate, come vedremo da il segno due punti (:).
Il C128 possiede due insiemi completi di caratteri che possono essere usati sia da tastiera che da programma. Vediamo una tabella riassuntiva dei caratteri escluso le lettere dell'alfabeto, i numeri e i segni grafici:

CARATTERE	DESCRIZIONE
BLANK	Separa le parole chiave ed i nomi delle variabili
;	Usato nelle variabili per la formattazione dell'output
=	Serve per assegnare un valore o per impostare una relazione
+	Serve per eseguire delle addizioni aritmetiche o per concatenare delle stringhe
-	Serve per delle sottrazioni
*	Moltiplicazione aritmetica
/	Divisione aritmetica
^	Elevazione a potenza
()	Servono come separatori di due espressioni
%	Definisce una variabile come INTERA
#	Precede il numero logico di un file ed e' utilizzato in istruzioni di input output
$	Definisce una variabile come STRINGA
,	E' usata nelle liste di variabili e separa anche i parametri di uno stesso comando
.	Punto decimale nelle costanti in virgola mobile
" "	Racchiudono costanti stringa
:	Separano piu' istruzioni basic su una stessa linea
?	Abbreviazione per il comando PRINT
<	Operatore relazionale
>	Operatore relazionale

P GRECO Costante numerica con valore 3.141592654

COSTANTI E VARIABILI

Le costanti sono i valori che si inseriscono nelle istruzioni Basic. Il programma tiene in memoria questi valori per descrivere ed utilizzare i dati durante l'esecuzione delle istruzioni. Il basic della Commodore puo' riconoscere ed elaborare tre tipi di dati:

1 NUMERI INTERI
2 NUMERI REALI
3 STRINGHE

Le costanti intere sono l'insieme dei numeri (senza punti decimali nel mezzo) e devono essere comprese fra -32768 e +32767. Se il segno piu' e' tralasciato la costante viene presa come numero positivo.
Si ricorda che non devono essere usati degli zeri perche' questi verranno comunque ignorati e l' unico effetto e quello di sprecare memoria rallentando l'esecuzione del programma.
Le costanti intere sono memorizzate come numeri binari di 2 bytes. Alcuni esempi, possono essere i seguenti.

```
    18
  3567
   -10
    99
     0
-32767
```

I numeri reali o costanti reali sono numeri positivi o negativi ed in questi sono compresi anche i frazionari.
Le parti decimali di un numero reale possono essere visualizzate adoperando il punto decimale. Si ricorda che le virgole non devono essere usate per separare la parte intera da quella decimale.
Le costanti reali si possono impiegare in due modi:

NOTAZIONE NORMALE

NOTAZIONE SCIENTIFICA

Le costanti reali vengono visualizzate sullo schermo fino alla nona cifra piu' il segno che se e' positivo sara' omesso.
Se si inseriscono piu' di 9 cifre il numero viene arrotondato alla decima cifra e se questa cifra e' un numero maggiore o uguale a 5 avviene un arrotondamento per eccesso.
I numeri reali vengono pero' trattati nei calcoli usando una precisione di 10 cifre e sono memorizzati usando 5 bytes di memoria. E' importante tenere conto di queste che potrebbero sembrare delle semplici osservazioni ma che invece devono fare parte del bagaglio di conoscenze del programmatore.
Alcuni esempi di numeri reali possono essere:

```
314
-12
 -.332211
```

I numeri piu' piccoli di .01 o piu' grandi di 999999999 sono trattati e quindi visualizzati in notazione scientifica. La notazione scientifica e' composta da 3 parti: MANTISSA, LETTERA, ESPONENTE.
La Mantissa e' un numero reale semplice, la lettera E viene usata per indicare la rappresentazione in forma esponenziale mentre l'esponente indica per quale potenza di 10 il numero deve essere moltiplicato.
Sia la mantissa che l' esponente possono essere numeri con segno positivo o negativo.
Il campo di definizione dell'esponente e' compreso fra -39 e +38 ed indica il numero di posizioni di cui il punto decimale nella mantissa sarebbe spostato a sinistra o a destra se il valore della costante fosse rappresentato come un numero semplice.
Tuttavia c'e' un limite anche per la grandezza dei numeri reali che il nostro computer e' in grado di manipolare. Infatti il numero piu' grande e':

+1.70141183 E +38

e quindi i calcoli che possono dare come risultato un numero maggiore di questo daranno un errore di:

?OVERFLOW ERROR

Il numero reale piu' piccolo e' invece:

+2.93873588 E -39

Ma i calcoli che daranno un risultato inferiore non riporteranno anche un messaggio di errore semplicemente porranno il risultato uguale a

zero.

COSTANTI STRINGA

Le costanti stringa sono gruppi di informazioni alfanumeriche come lettere, numeri, simboli grafici.
Quando si digita una stringa da tastiera la sua lunghezza non puo' superare lo spazio disponibile sulla linea di programma.
Una costante stringa puo' contenere spaziature, lettere, numeri punteggiature e caratterri di controllo cursore in qualsiasi combinazione. Si possono anche inserire virgole fra i numeri mentre l'unico carattere che non puo' essere inserito in una stringa e' il doppio apice o virgolette (") perche' questo carattere viene impiegato per definire da che punto incomincia e a che punto finisce la stringa.
Una stringa puo' assumere anche un valore nullo cioe' a dire non contenere alcun carattere. Questo viene fatto con l'apertura e la chiusura immediata delle virgolette.

"" Stringa nulla

Si puo' trascurare anche la chiusura delle virgolette al termine di una stringa se questa e' l'ultima parte di una linea di programma o se e' seguita dai due punti (:).
Per includere nelle stringhe le virgolette si usa CHR$(34).

LE VARIABILI

Le costanti sono dati fissi ma un programma non puo' basarsi solo su informazioni immutabili. Per questo impiegheremo le variabili che come dice la parola stessa possono variare il loro valore durante l' elaborazione. Oppure in fase di input possono accettare i valori che daremo.
Infatti il valore rappresentato da una variabile puo' essere assegnato ponendo questa uguale ad una costante oppure puo' essere il risultato dei calcoli del programma.
I dati delle variabili possono essere, come abbiamo gia' visto per le costanti in precedenza, numeri interi, reali o stringhe.
I nomi delle variabili possono avere una lunghezza qualsiasi ma solamente i primi due caratteri possono essere riconosciuti dal Basic Commodore. Cioe' a dire che tutti i nomi usati per variabili diverse non devono avere i primi due caratteri uguali. Inoltre i nomi delle variabili non possono essere ne' possono contenere parole riservate del Basic.
Queste parole riservate includono sia i comandi che le istruzioni, i nomi di funzione ed i nomi degli operatori logici.
I caratteri usati per formare i nomi delle variabili sono quelli dell'alfabeto ed i numeri da 0 a 9 con la regola che il primo carattere del nome della variabile deve essere OBBLIGATORIAMENTE una lettera.
I caratteri di dichiarazione del tipo di dato inserito nella variabile devono essere usati come ultimo carattere identificatore della variabile stessa. Avremo questi tipi di variabili:

A - Variabile intera che avra' come segno di

identificazione il simbolo %

B - Variabile stringa che avra' come segno di identificazione il simbolo $

C - Variabile reale che non avra' nessun simbolo.
Vediamo alcuni esempi:

AA=2.7 variabile reale
AA%=3 variabile intera
AA$="MILANO" variabile stringa

ESPRESSIONI ED OPERATORI

Le espressioni sono formate usando costanti, variabili e MATRICI che abbiamo visto a proposito del comando DIM.
Una espressione puo' essere composta da una singola costante o puo' anche essere una combinazione di costanti e variabili con operatori aritmetici, relazionali o logici, definiti per avere uun determinato risultato. Le espressioni possono essere distinte in due classi:

ARITMETICHE
STRINGA

Le espressioni aritmetiche quando vengono risolte danno come risultato un valore intero o reale.
Gli operatori aritmetici dovrebbero essere abbastanza noti perche' servono per eseguire addizioni, sottrazzioni, moltiplicazioni,

divisioni ed elevamenti a potenza.
Un'altra categoria di operatori e' chiamata **OPERATORI RELAZIONALI** e benche' siano usati per confrontare i valori di due operandi producono anche risultati aritmetici.
Quando si usano in una operazione di confronto gli operatori relazionali e gli operatori logici danno la valutazione VERO/FALSO di una espressione.
In una espressione se il rapporto e' vero al risultato viene assegnato il valore -1 mentre se e' falso il risultato e' 0. Gli operatori relazionali sono i seguenti:

<   Minore
=   Uguale
>   Maggiore
<=   Minore o uguale
>=   Maggiore o uguale
<>   Diverso

Gli operatori relazionali possono essere utilizzati per il confronto di stringhe. In questo caso per l'ordinamento delle lettere dell'alfabeto e' preso come prima lettera A, seconda lettera B, terza lettera C ecc... tenendo presente che il computer utilizza l'alfabeto inglese.
I dati numerici possono essere confrontati solamente con altri dati numerici cosi' come le stringhe possono essere confrontate solamente con altre stringhe. Nel caso non si segua questa prescrizione verra' generato un messaggio di errore:

?TYPE MISMATCH

Gli operatori logici possono essere usati per modificare i significati degli operatori relazionali o per produrre un risultato aritmetico.
Gli operatori logici chiamati anche operatori BOOLEANI possono essere anche usati per eseguire operazioni logiche su due operandi di cui viene considerata una sola cifra binaria.
Vediamo ora una tavola Booleana della verita':

1 AND 1=1
0 AND 1=0
1 AND 0=0
0 AND 0=0

Il risultato dell'operazione AND e' 1 solo se entrambi i bit sono uguali a 1.

1 OR 1=1
0 OR 1=1
1 OR 0=1
0 OR 0=0

Il risultato della OR e' 1 solo se almeno un bit e' uguale a 1.

NOT 1 = 0
NOT 0 = 1

esegue il complemento logico di ciascun bit

1 XOR 1 =0
1 XOR 0 =1
0 XOR 1 =1
0 XOR 0 =0

La OR esclusiva (XOR) non ha un operatore logico riconosciuto ma viene eseguita come parte dell'istruzione WAITE. La OR esclusiva significa che se i bit dei due operandi sono uguali allora il risultato e' 0 altrimenti e' 1.

ORDINE GERARCHICO

Nell' escuzione di operazioni il computer osserva, e non potrebbe essere altrimenti una serie di precedenze per cui una determinata operazione sara' eseguita prima di un' altra. Non e' una procedura difficile da comprendere ne' da osservare, vediamo come.
Le parentesi chiuse ed aperte fanno si che i valori al loro interno siano una sola espressione. Il loro uso e' quello tipico della matematica mentre .dobbiamo ricordare che nel basic si usa un solo tipo di parentesi, quelle tonde (), ripetute piu' volte. Ci devono essere tante parentesi chiuse quante ne sono state aperte e le operazioni avvengono a partire dalla coppia piu' interna di parentesi.
Come in matematica cosi' nel basic esiste un ordine gerarchico nell' esecuzione delle operazioni.
L'ordine gerarchico degli operatori ed il modo operativo e' il seguente:

1) Per prima viene fatta la moltiplicazione seguita dalla divisione poi dalla somma e infine dalla sottrazione.

2) Le espressioni all'interno delle parentesi sono calcolate per prime partendo dalla prima coppia di parentesi, cioe' da quella piu'

interna. Facciamo un esempio:

```
?(4+8)/2

        6
```

In questo caso per prima e' stata eseguita la somma all'interno delle parentesi e quindi il risultato diviso per 2.
Nel caso non si fossero utilizzate le parentesi si sarebbe giunti ad un diverso risultato. Infatti:

```
?4+8/2

        8
```

perche' in questo caso prima sarebbe stata eseguita la divione e poi la somma con il risultato di 8 invece che di 6.
Ricordiamo che il punto esclamativo sta per i comando PRINT.
Inserendo dentro le parentesi gruppi di operandi racchiusi essi stessi fra parentesi si possono formare delle espressioni a piu' livelli. Questo procedimento prende il nome di **NIDIFICAZIONE**.
Le parentesi possono essere nidificare fino ad un massimo di 10 livelli per espressione.

OPERAZIONI SU STRINGHE

Le stringhe vengono conforontate utilizzando gli stessi operatori relazionali (maggiore, minore, uguale ecc...) impiegati per i numeri. I confronti di stringhe vengono fatti prendendo un

carattere alla volta da sinistra a destra di ciascuna stringa e valutando il codice del carattere prelevato.
Se i codici carattere sono uguali allora anche i caratteri sono considerati uguali. Se i codici carattere sono diversi allora il carattere con il numero di codice piu' basso risulta minore. I confronti hanno termine quando viene raggiunta la fine dell'ubna o dell'altra stringa. Se tutti i codici sono uguali viene considerata minore la stringa piu' corta.
E' importante ricordare che gli spazi bianchi all'interno, all'inizio o alla fine di una stringa sono consideraticome significativi, cioe' hanno un valore che e' appunto quello del carattere BLANK.
Alla fine di tutti i confronti, indipendentemente dal tipo dei dati impiegati si ottiene un risultato intero.
Questo e' vero anche se entrambi gli operandi sono stringhe. Inoltre il confronto fra due operandi stringa puo' essere adoperato come operando nell'esecuzione di calcoli.
Le espressioni stringa sono trattate come se fossero seguite dalla parola o dai simboli corrospondenti diverso da 0.
Cio' significa che se una espressione risulta vera verranno eseguite le istruzioni che seguono l'espressione stessa sulla linea di programma.
Se invece l'espressione e' falsa il resto della linea e' ignorato e viene eseguita la linea di programma successiva.
L'unico operatore aritmetico riconosciuto dal basic per le operazioni sulle stringhe e' il segno piu' (+) usato per eseguire la concatenazione di due o piu' stringhe.
Abbiamo visto nel capitolo precedente che esistono una serie di comandi per operare estrazioni e che quindi rendono in pratica disponibile anche una operazione come la

sottrazione.
Quando si concatenano due o piu' stringhe la stringa alla destra del segno piu' viene aggiunta alla stringa che si trova alla sinistra dello stesso segno e che quindi potra' essere stampata come terza stringa oppure usata per determinate manipolazioni o assegnata ad una variabile.
Ricordiamo che nessuna stringa anche frutto di concatenazione fra altre puo' essere maggiore di 255 caratteri compresi in questo gli spazi bianchi.

CAPITOLO TERZO

LA GRAFICA

Il BASIC 7.0 del modo C128, mette a disposizione una notevole serie di risorse per la gestione grafica.
Come abbiamo detto nell' introduzione, cercheremo di esaminare le caratteristiche di queste funzioni senza pretendere di scrivere un trattato sulla grafica.
Inoltre non staremo a ripetere la descrizione dei singoli comandi passo passo perche' questi sono stati visti abbastanza in dettaglio nelle precedenti pagine relative appunto ai comandi.
Per prima cosa ricordiamo che sono disponibili due formati di schermo entrambi controllati da due integrati messi a punto proprio dalla COMMODORE per l'elaborazione della grafica.
Il primo CHIP e' il vecchio e famoso VIC o VIDEO INTERFACE CONTROLLER che controlla il formato di schermo a 40 colonne. Di questo integrato e' stato parlato diffusamente fin dalla sua nascita e dal suo impiego sul vecchio VIC 20.
Questo integrato oltre a controllare il modo 40 colonne gestisce anche sedici colori e l' alta risoluzione grafica che infatti e' teoricamente almeno disponibile solo sulle 40 colonne.
L' altro integrato e' l' 8563 VDC che controlla e gestisce il formato video ad 80 colonne, 16 colori ed un modo di visualizzazione schermo per i caratteri e una certa grafica.
Nella prima parte di questo capitolo vedremo una spiegazione dei comandi grafici, del Modo Alta risoluzione, Multicolor e Split SCREEN e nella

seconda parte cercheremo di trattare approfonditamente gli Sprites o OGGETTI MOBILI della grafica.
Per prima cosa vediamo l' insieme dei comandi dedicati alla grafica che abbiamo a disposizione facendone una tabella riassuntiva e cercando di approfondirne il funzionamento.

I COMANDI GRAFICI

BOX = Serve per disegnare un poligono sullo schermo in Alta risoluzione o BIT MAP.

CHAR = Serve per visualizzare i caratteri sempre in Alta risoluzione.

CIRCLE = Serve per disegnare cerchi, ellissi ed altre figure geometriche che richiedono delle parti curve.

COLOR = Con questo comando si possono selezionare tutte le combinazioni di colori relative a carattere, interno, esterno schermo e bordo.

DRAW = Serve per diegnare punti o linee, viste queste come insieme di punti, sullo schermo sempre che si sia in Alta risoluzione.

GRAPHIC = Seleziona il tipo di schermo o MODO su cui si desidera operare. Questi come vedremo potra' essere Alta Risoluzione, Multicolore, Split SCREEN ecc.

PAINT = Riempe una scelta area di schermo con un colore. Opera in Alta risoluzione.

SCALE = Fissa la scala delle immagini in modo Alta risoluzione.

SPRITE = Definisce uno Sprite

SPRSAV = Serve per l' immagazzinamento degli Sprites.

SSHAPE = Immagazzina una parte dello schermo in Alta risoluzione.

La prima cosa da scegliere e il tipo di schermo sul quale si desidera lavorare. Vedremo poi il colore e che cosa disegnare.
Quando accendiamo il computer a secondo se il tasto delle colonne e' premuto o no ci troviamo in modo 40 (tasto alto) o in modo 80 colonne (tasto o interruttore premuto).
Per cambiare modo di visualizzazione grafica o MODO GRAFICO si usa il comando :

GRAPHIC n

In cui assegnando a n uno dei seguenti valori avremo la scelta di un MODO GRAFICO. Vediamo la tabella seguente:

MODO	DESCRIZIONE
0	40 colonne testo
1	Alta risoluzione
2	Alta risoluzione SPLIT SCREEN
3	Multicolore
4	Multicolore SPLIT SCREEN
5	80 colonne testo.

Se andate a rivedere la descrizione di questo comando nelle precedenti pagine noterete che ci sono due parametri opzionali che servono per delimitare, nel modo SPLIT SCREEN, la dimensione della finestra oltre alla possibilita' di usare il CLEAR per la pulizia del tipo di schermo scelto.
Approfondiamo un po' il motivo di CLR.
Quando si seleziona un MODO grafico il computer riserva una parte cospicua della sua memoria, esattamente 9K Bytes per contenere i dati relativi all' alta risoluzione (8K) e per i dati dei colori (1K).
Per questo motivo ci da immediatamente la possibilita' con un comando di riorganizzare la memoria mettendola a disposizione per altri disegni.
Vediamo un po' di approfondimento sui MODI GRAFICI.

MODO TESTO STANDARD

Come abbiamo detto e' il modo in cui ci troviamo all' accensione. Supponiamo sempre di essere su 40 colonne che e' dove NORMALMENTE si opera in grafica.
In questa modalita' possono essere al massimo visualizzati 1000 caratteri (25 righe di 40 caratteri ciascuna).
Ciascun carattere viene rappresentato da una matrice di 8x8 punti in uno qualsiasi dei 16 colori disponibili.
I caratteri, o meglio i valori che li definiscono, possono essere prelevati da ROM o da RAM, anche se solitamente sono da ROM.
Quando per un programma si desiderano particolari caratteri grafici tutto cio' che e'

necessario fare e' definire su RAM le nuove forme di carattere tramite una serie opportuna di valori e comunicare all' integrato VIC che li controlla di andare a prelevare le informazioni sui caratteri da li anziche' dalla ROM dove normalmente dirigeva la sua richiesta informativa.
Per visualizzare poi i caratteri a colori il VIC accede alla memoria di schermo per determinare il codice carattere per quella locazione di schermo.
Allo stesso tempo accede alla memoria colore per determinare quale colore si desidera per quel determinato carattere.

MODO ALTA RISOLUZIONE

Il modo Alta Risoluzione viene utilizzato per creare grafica al massimo grado di precisione consentito. In questo modo lo schermo viene visto come una grande griglia con 320 punti in orizzontale e 200 in verticale.
Con questo modo di operare abbiamo quindi il controllo sui piccoli punti dello schermo che vengono chiamati PIXEL, acronimo di PICTURE CELL.
Ogni punto della memoria schermo puo' assumere due valori: 1 (UNO) per acceso e 0 (ZERO) per spento o trasparente.
Se il punto e' acceso questo verra' colorato con il colore del carattere che si e' scelto per quella determinata posizione di schermo.
Quando si usa il modo di caratteri in Alta Risoluzione tutti i punti interni ad una griglia 8X8, che e' quella che abbiamo visto per il carattere possono avere sia il colore di fondo che il colore principale.

Cio' limita in qualche modo la risoluzione del colore all'interno dello spazio. Difatti possono sorgere dei problemi quando si incrociano linee di differenti colori. Il modo multicolore che vedremo dopo risolve questo problema.
Il VIC e' stato progettato perche' l'Alta risoluzione sia disponibile per mezzo della scanzione BIT MAP dello schermo. Con questo metodo e' possibile assegnare ad ogni punto di risoluzione o PIXEL un bit in memoria.
Il modo di grafica ad alta risoluzione presenta alcuni inconvenienti che spiegano il motivo per cui raramente viene usato in pieno.
Innanzi tutto la scansione punto per punto dell'intero schermo richiede una notevole quantita' di memoria in quanto un simile controllo richiede un bit di memoria per ogni pixel.
Poiche' ogni carattere e' una matrice di 8x8 e ci sono 25 linee di 40 caratteri l'una la risoluzione per l'intero schermo e' data da 320 pixel per 200 pixel per un totale di 64000 punti distinti ognuno dei quali richiede un bit di memoria.

MODO MULTICOLORE

Con il modo multicolore la risoluzione orizzontale viene sacrificata alla possibilita' di usare piu' colori.
Lo schermo in questo caso, risultera' composto da 160 punti orizzontali per 200 verticali. Ogni punto orizzontale e' quindi piu' ampio che nel modo visto in precedenza. Il modo multicolor permette di usare sino a 4 colori differenti ma limitatamente ad ogni zona di 8x8 punti.
Quindi il problema visto in precedenza

dell'incrocio di piu' linee di differenticolori puo' essere risolto in questo modo. L'unico sacrificio e' relativo alla risoluzione orizzonatale che si dimezza in quanto ogni punto del modo multicolore e' largo il doppio di un punto ad alta risoluzione. Questa minima perdita di risoluzione e' largamente compensata dalle capacita' del modo multicolore.

MODO SPLIT SCREEN

Il modo SPLIT SCREEN riunisce insieme i modi alta risoluzione e testo standard.
Nel modo split screen la parte superiore dello schermo e' in alta risoluzione con 320 punti orizzontali e 160 verticali.
La parte inferiore dello schermo e' una FINESTRA dove e' possibile visualizzare 5 o piu' linee di testo.

SELEZIONE COLORI

Sul Commodore C128 e' possibile impostare indipendentemente l'area dello sfondo del bordo e dell'interno su uno qualsiasi dei 16 colori disponibili.
Si possono anche impostare due registri multicolor anche se cio' ha effetto solo quando si usa il multicolor.
Per selezionare i colori impiegheremo il comando:

COLOR sorgente, colore

in cui sorgente e' la parte dello schermo che si desidera colorare (sfondo, interno, bordo ecc...), colore e' il codice del colore.
Per quanto riguarda il parametro sorgente vediamo che valori puo' assumere:

VALORE	SORGENTE
0	Colore di fondo su 40 colonne
1	Colore esterno per schermo grafico
2	Colore 1 esterno per schermo in multicolor
3	Colore 2 esterno per schermo in multicolor
4	Colore del bordo su 40 colonne sia in grafica che testo
5	Colore del carattere per testo a 40 o 80 colonne
6	Colore fondo per schermo a 80 colonne

Vediamo ora i codici numerici relativi ai colori per lo schermo a 40 colonne e per lo schermo ad 80 colonne.

CODICI COLORE PER SCHERMO A 40 COLONNE

CODICE	COLORE
1	Nero
2	Bianco
3	Rosso
4	Cian
5	Porpora
6	Verde
7	Bleu
8	Giallo
9	Arancio

10	Marrone
11	Rosso chiaro
12	Grigio scuro
13	Grigio medio
14	Verde chiaro
15	Bleu chiaro
16	Grigio chiaro

CODICI COLORE PER SCHERMO A 80 COLONNE

CODICE	COLORE
1	Nero
2	Bianco
3	Rosso scuro
4	Cian chiaro
5	Porpora chiaro
6	Verde scuro
7	Bleu scuro
8	Giallo chiaro
9	Porpora scuro
10	Giallo scuro
11	Rosso chiaro
12	Cian scuro
13	Grigio medio
14	Verde chiaro
15	Bleu chiaro
16	Grigio chiaro

Per controllare quali colori siano stati impostati per ultimi si usa la funzione RCLR (N) che riporta il colore assegnato a quella sorgente.

VISUALIZZAZIONE DI DISEGNI SULLO SCHERMO

Abbiamo visto come effettuare la selezione dei tipi di schermo sui quali si vuole operare ed i colori che si vogliono impiegare.
Prima di iniziare a disegnare e' importante comprendere un concetto fondamentale della grafica: il PIXEL CURSOR o PC.
Il PC e' simile al cursore mobile che si puo' vedere nel modo carattere standard che indica dove verra' visualizzato il successivo carattere.
Anche se e' invisibile, il PC indica dove verra' disegnato il prossimo punto sullo schermo. E cio' sia in alta risoluzione sia in multicolor.
Infatti nei comandi grafici in cui sono state messe le coordinate o i parametri relativi opzionali, il PC e' usato come indicatore di coordinate di DEFAULT.
Il comando LOCATE permette di collocare il PC in qualsiasi punto dello schermo. Naturalmente i risultati del comando LOCATE non verranno visualizzati sino al momento in cui non sara' effettivamente disegnato qualcosa.
Il comando LOCATE ha 2 parametri X e Y in cui X rappresenta la distanza orizzontale attraverso lo schermo espressa in numero di punti. Quando X e' uguale a 0 vorra' dire che il PC e' sull'estremo margine sinistro dello schermo.
Y rappresenta la distanza verticale lungo lo schermo sempre espressa in numero di punti. Quando Y e' uguale a 0 il PC e' sul margine superiore dello schermo.
Come abbiamo detto se una coordinata opzionale X o Y e' omessa la posizione del PC attuale e' usata come posizione di DEFAULT.
Le coordinate X e Y possono essere specificate sia come valore assoluto sia come scarto dalla attuale posizione del PC.

Facendo precedere il valore X o Y da un segno piu' o meno (+,-) il PC verra' mosso in una direzione positiva o negativa relativamente alla sua posizione attuale tenendo presente che ci troviamo in un sistema di assi cartesiane la cui origine e' fissata al margine superiore sinistro dello schermo. Assi cartesiane in pratica rovesciate rispetto al normale.
Percio' il segno piu' (+) prima del valore X muove il PC verso destra ed il segno meno (-) muove il PC verso sinistra.
Allo stesso modo il segno piu' (+) davanti al valore Y muove il PC verso il basso rispetto alla sua attuale posizione, e il segno meno (-) lo muove verso l'alto.
Dovunque si definisca esplicitamente una coordinata X o Y nei comandi grafici, possono essere utilizzati sia valori assoluti che scarti relativi. Vediamo alcuni esempi.

LOCATE 0,0

Pone il PC sull'origine delle assi cartesiane rovesciate che sono l'angolo superiore sinistro dello schermo.

LOCATE 160,100

Pone il PC nell'esatto centro dello schermo quando siamo in modo alta risoluzione.

LOCATE -20,+30

Sposta il PC di 20 punti a sinistra e di 30 punti verso il basso

In questi esempi le posizioni del PC sono state date sia come valore assoluto sia come scarto relativo.
Nel comando LOCATE come in parecchi altri

utilizzati nella grafica si puo' utilizzare un modo alternativo per esprimere una nuova posizione. Infatti oltre alla distanza puo' essere determinato anche un angolo relativo rispetto all'attuale posizione del PC usando un punto e virgola al posto della virgola. Ad esempio:

LOCATE 30;90

Il PC in questo caso viene spostato dalla sua attuale posizione per una distanza di 50 pixel ad un angolo di 90 gradi.
Si puo' conoscere in qualsiasi momento la posizione esatta del PC usando la funzione RDOT la quale fornira' la posizione attuale del PC relativa alle sue coordinate o la sorgente colore dell'attuale posizione.

LE FIGURE ED I DISEGNI

Dopo aver visto come scegliere il tipo di schermo sul quale operare e come si fa a scegliere i colori vediamo qualche rappresentazione grafica o disegno di figure.
Non staremo a rispiegare i singoli comandi che sono stati abbondantemente trattati nella parte precedente.
Il comando CIRCLE ad esempio disegna i cerchi o comunque figure curve con un'insieme di rette.
Questa operazione viene fatta calcolando il punto successivo sulla circonferenza e disegnando poi una linea retta dal punto precedente. Infatti il parametro INCREMENTO che gfa parte del comando e il cui valore di DEFAULT e' di 2 gradi specifica di quanti gradi il PC venga mosso in senso orario tracciando quindi la

linea successiva. Aumantandone il valore relativo avremo un contorno del cerchio sempre piu' grossolano fino al punto in cui il cerchio apparira' come un poligono regolare. Infatti:

```
CIRCLE,260,40,20,,,,,90
```

Disegnera' un rombo, mentre:

```
CIRCLE,60,140,20,18,,,120
```

disegnera' un triangolo
Per avere un cerchio al centro dello schermo daremo:

```
CIRCLE 1,150,130,40,40
```

Altro comando importante, anche per eventuali scopi didattici e' il PAINT che ci consente di riempire una determinata area di schermo limitata.
La colorazione puo' iniziare sia dall' attuale posizione del PC oppure dalle coordinate specificate. Come al solito alle coordinate puo' essere dato sia un valore assoluto che relativo. Prendendo l' esempio precedente relativo al cerchio al centro dello schermo, questo potra' essere riempito:

```
CIRCLE 1,150,130,40,40:PAINT1
```

Ricordiamo ancora una volta di aver selezionato il modo grafico appropriato ed il colore.

NOTA

E' probabile che qualche esempi, una volta inserito nel computer non ottenga nessun effetto VISIBILE. Si consiglia di cambiare i colori in funzione anche del tipo di Monitor o TV o

provenienza del computer stesso.

Vediamo ora un esempio di disegno di una corona circolare o cerchi concentrici:

```
10 GRAPHIC3,1
20 COLOR2,2
30 CIRCLE2,80,100,40,60
40 CIRCLE2,80,100,20,30
50 PAINT2,100,130
60 GETKEY A$
70 GRAPHIC 0
```

Altro comando che puo' essere utilimente impiegato per disegnare forme geometriche e il BOX che puo' essere anche questo ruotato e colorato internamente come dall' esempio seguente:

```
BOX,10,10,60,60,45,1
```

VISUALIZZAZIONE CARATTERI

Il comando CHAR permette di utilizzare contemporaneamente il modo di visualizzazione testo ed una visualizzazione in Alta risoluzione.
Infatti contrariamente al PRINT che puo' essere usato solo nel modo TESTO o nella parte bassa del modo SPLIT SCREEN, con CHAR si puo' disporre il testo in qualsiasi punto dello schermo in qualunque modo.
Un' altra particolarita' della grafica su questo computer e' la possibilita' di disporre di un

comando SCALE.
Questo comando infatti offre la possibilita' di allargare o ridurre le dimensioni del disegno sullo schermo.
Come sappiamo, nel modo Alta Risoluzione lo schermo a 40 colonne ha 320 punti di coordinate orizzontali e 200 verticali, mentre nel modo Multicolore, dato che la risoluzione orizzontale si dimezza avremo 160x200.
Ricordiamo che questa diminuzione della risoluzione e' compensata dalla capacita' di utilizzare un colore addizionale per un totale di 3 colori sempre all' interno della matrice di 8x8 del carattere.
Quando si usa il comando SCALE in entrambi i modi avremo delle coordinate proporzionali l' una all' altra.

NOTA CONCLUSIVA

Al di la' dei singoli comandi che gia' consentono notevoli applicazioni, per sfruttare appieno le possibilita' grafiche sia del VIC che del 8563 VDC e' necessario conoscere ed operare in Linguaggio Macchina o almeno comprenderne le implicazioni. E' quindi indispensabile accedere al Sistema Operativo del computer.
A parte quanto e' scritto in questa guida il rapporto al Linguaggio Macchina (vedi i successivi capitoli) la EVM Computers sta mettendo a punto un manuale completo sull' argomento.

GLI SPRITES

Una delle caratteristiche piu' interessanti del C128 sia quando opera in MODO 128 che in MODO 64, e' la sua capacita' di visualizzare oggetti mobili chiamati SPRITE.
Gli sprite sono delle immagini grafiche che si possono definire come forma e colore e collocare in qualsiasi punto all'interno dello schermo o fuori dello stesso. Gli sprite, chiamati anche MOVABLE OBJECT, sono particolarmente adatti per animazioni a scopo grafico o per giochi.
In ogni momento si possono visulizzare sullo schermo fino a 8 sprite ed e' inoltre possibile definire ogni sprite sia con una figura in alta risoluzione, sia in multicolor.
Inoltre ciascuna figura puo' essere espansa nella direzione dell'asse X orizzontale e/o in quella delle assi Y o verticale.
Gli sprites possono anche essere combinati tra di loro per creare immagini grafiche piu' grandi e colorate.
Ad ogni sprite puo' essere assegnata una priorita' di visualizzazione che fara' apparire il suo movimento come se avvenisse davanti o dietro alle immagini di un' altra visualizzazione grafica.
Questa caratteristica permette di creare un effetto grafico tridimensionale.
Inoltre saremo anche in grado di rilevare quando un qualsiasi sprite entra in collisione con un altro o con una forma presente comunque sullo schermo.
Non staremo qui a parlare approfonditamente del sistema di utilizzo degli sprites nel modo 64 perche' rimandiamo alla relativa guida e anche perche' si tratta del sistema piu' difficile per impiegare gli sprites dovendo utilizzare un

insieme di comandi POKE su dati indirizzi di memoria, cosa abbastanza difficile.
Comunque vediamo in breve quali sono i sistemi per creare gli sprites. Nel modo 128 si possono creare gli sprites in queste 3 diverse maniere:

1 Utilizzando i comandi sprite entro un programma;

2 Utilizzando il modo di definizione sprite con il comando SPR DEF.

3 Utilizzando lo stesso sistema che si impiega nel Commodore 64.

Tralascindo il terzo modo che come abbiamo detto e' il piu' complesso spieghiamo i due modi tipici del CBM 128 partendo dal primo.

UTILIZZO DEI COMANDI SPRITE DI UN PROGRAMMA.

Questo primo sistema consiste come si puo' vedere dallo schema riportato in un metodo piuttosto semplice ma allo stesso tempo molto approfondito e molto facile per creare degli sprites. In altre parole si tratta di disegnare una figura utilizzando i comandi grafici visti in precedenza quindi di memorizzarla in una stringa e poi di memorizzare la stringa stessa in un'area sprite e successivamente di assegnargli un movimento.
Per evidenziare al meglio quanto detto proviamo a scrivere un piccolo programm (ripreso dal manuale interno CBM), sul quale si consiglia poi di apportare tutte quelle variazioni che portano ad una maggiore dimestichezza con il problema.

```
5 COLOR 0,1:COLOR 4,1
10 GRAPHIC 1,1
15 BOX 1,2,2,45,45
20 DRAW 1,17,10 TO 28,10 TO 26,30  TO  19,30  TO
17,10
22 DRAW 1,11,10 TO 15,10 TO 15,18  TO  11,18  TO
11,10
24 DRAW 1,30,10 TO 34,10 TO 34,18  TO  30,18  TO
30,10
26 DRAW 1,11,20 TO 15,20 TO 15,28  TO  11,28  TO
11,20
28 DRAW 1,30,20 TO 34,20 TO 34,28  TO  30,28  TO
30,20
30 DRAW 1,26,28 TO 19,28
32 BOX 1,20,14,,26,18,90,1
35 BOX 1,150,35,195,40,90,1
37 BOX 1,150,135,195,140,90,1
40 BOX 1,150,215,195,220,90,1
42 DRAW 1,50,180  TO  300,180:DRAW  1,50,180  TO
50,190:DRAW 1,300,180 TO 300,190
43 DRAW 1,50,190 TO 300,190
44 CHAR 1,18,23"ARRIVO"
```

A questo punto noi abbiamo creato la figura di un' auto in un box nella parte alta dello schermo. Avremo inoltre disegnato un pista con la linea di arrivo.
Il secondo passo sara' di memorizzare con un apposito comando i dati relativi a questa figura all'interno di una stringa. Impiegheremo per questo il comando SSHAPE:

```
45 SSHAPE A$,10,11,34,31
```

Questo comando immagazzina l'immagine dello schermo all'interno di una variabile stringa per successive elaborazioni in rapporto alle quattro

coordinate dello schermo che noi gli abbiamo dato.
I numeri 10, 11, 34, 31 sono le coordinate del disegno. Dovete assegnare a queste coordinate i valori giusti oppure il disegno sara' memorizzato male all'interno della stringa.
Infatti se posizionate il comando SSHAPE su una zona vuota dello schermo questa stringa di dati sara' vuota. Percio' quando andrete a trasferire questi valori all'interno di uno sprite non verra' trovato alcun dato. Ricordate anche che il disegno da memorizzare nella stringa non deve essere superiore ai 24 punti di larghezza per 21 di altezza che sono le dimensioni di ogni singolo sprite.
Il comando SSHAPE trasferisce il disegno all'interno di una stringa di dati che il computer interpreta come dati di disegno. La stringa di dati A$ immagazzina una serie di 0 e di 1 nella memoria del computer che equivalgono direttamente al disegno sullo schermo.
Il sistema di memorizzazione grafica dei dati e' abbastanza simile in quasi tutti i computer e si riduce, semplificando a questo metodo.
Ogni punto sullo schermo chiamato pixel ha un corrispondente bit nella memoria del computer che lo controlla. Nel modo BIT-MAP standard se il bit e' a 1 cioe' attivo (ON) allora il pixel relativo sara' attivo cioe' acceso. Avremo in altre parole un' immagine luminosa per quel punto. Invece se il bit in memoria e' a 0 allora il pixel e' spento cioe' non avremo nessuna immagine luminosa e avremo solo la trasparenza.

CARICAMENTO DI UN DISEGNO SU UNO SPRITE

Il disegno e' ora immagazzinato in una stringa.

Il successivo passo e' di trasferire il disegno o meglio l'insieme di dati che costituiscono il disegno e che formano quindi la stringa entro un'area chiamata SPRITE DATA AREA in maniera tale che ci si possa lavorare sopra e si possa quindi animare. Il comando necessario per eseguire questa operazione e' il comando SPRSAV. Vediamone l'applicazione:

```
10 SPRSAV B$,5
55 SPRSAV B$,6
```

In questo modo i dati relativi al disegno sono stati trasferiti nell'area dello sprite 5 e nell'area dello sprite 6.
Entrambi gli sprites hanno gli stessi dati e per questo motivo potra' essere piu' semplice successivamente cambiare o alterare l'uno o l'altro.

ATTIVAZIONE DEGLI SPRITES

Il comando SPRITE attiva un dato sprite che potra' essere da 1 a 8, lo colora, ne specifica le priorita' sullo schermo, ne espande le dimensioni e determina il tipo di visualizzazione dello sprite stesso. Come abbiamo accennato in precedenza la priorita' dello sprite si riferisce al fatto che lo sprite puo' passare davanti o dietro a un oggetto sullo schermo. Inoltre gli sprites possono essere espansi cioe' allargati o allungati il doppio delle dimensioni originali. Il tipo di visualizzazione sprite si riferisce invece al fatto che questi puo' essere in modo BIT-MAP standard oppure in modo MULTICOLORE.

Vediamo ora 2 esempi applicativi del comando sprite in cui si attivano gli sprites 5 e 6.

```
100 SPRITE 5,1,7,0,0,0,0
110 SPRITE 6,1,3,0,0,0,0
```

I parametri del comando SPRITE sono i seguenti:

SPRITE #,O,C,P,X,Y,M

= numero dello sprite sul quale si vuole operare (1-8)

O = attiva o disattiva e quindi O=1 attiva, O=0 disattiva

C = colore (vedere le tabelle dei colori)
P = la priorita'. Se P=0 lo sprite e' davanti all'oggetto sullo schermo; se P=1 lo sprite e' dietro all'oggetto dello schermo

X = espansione in orizzontale. Se X=1 lo sprite si espande nella direzione orizzontale; se X=0 lo sprite e' nella dimensione normale

Y = espansione in verticale. Se Y=1 lo sprite si espande in direzione verticale; se Y=0 lo sprite e' in dimensione normale

M = Se M=1 lo sprite e' multicolor; se M=0 avremo la visualizzazione di uno sprite standard.

Da notare quindi che il comando SPRITE e' estremamente potente perche' ci da il controllo sopra tutte le numerose e varie qualita' di uno

sprite.

COME MUOVERE GLI SPRITES

Una delle piu' interessanti e potenti caratteristiche di questa versione del Basic e' la possibilita' di posizionare e animare gli sprites.
Per ogni sprite e' possibile usare il comando MOVSPR per regolarne la posizione, farlo muovere e fermarlo. La sintassi del comando MOVSPR e' la seguente:

```
MOVSPR NUM,X1,Y1
```

dove NUM e' il numero dello sprite da 0 a 7 la cui posizione si voglia impostare o cambiare. La coordinata X1, Y1 e' la nuova locazione dello sprite.
Per posizionare lo sprite si puo' stabilire X1, Y1 come valori assoluti o come scarti relativi. Utilizzando uno scarto relativo per posizionare uno sprite, bisogna essere consapevoli che la nuova posizione viene calcolata dalla posizione corrente dello sprite piuttosto che dalla locazione corrente del PC.
Gli sprites vengono posizionati rispetto al loro angolo superiore di sinistra. Vi e' una posizione specifica dello schermo detta "finestra" in cui sono visibili gli sprites. Le coordinate che definiscono i limiti dello schermo bit-map in altre parole. Per esempio nell'angolo superiore dello schermo bit-map e' 0,0. Per posizionare l'angolo superiore sinistro di uno sprite queste coordinare saranno 24,50. Si dovra' usare una apposita tabella che noi

abbiamo riportato nel comando SCALE, per calcolare sia le coordinate assolute di posizione sia le relative distanze di movimento per gli sprites. Per animare uno sprite e per fermarlo si puo' anche usare una forma speciale del parametro X1, Y1 del comando MOVSPR. In questo caso le parti X1 Y1 del parametro sono separate da un # invece che da una ,. Il valore X1 specifica in gradi per la direzione in cui si muovera' lo sprite un angolo che si forma in senso orario. Il valore Y1 e' un numero da 0 a 15 e specifica una velocita' costante dello sprite dove 0 ferma il movimento dello sprite e 15 ne rappresenta invece la velocita' massima. La velocita' dello sprite si misura dal numero dei punti che lo sprite supera in un certo periodo di tempo nella direzione indicata. Il movimento dello sprite e' in realta' una serie di scarti relativi istantanei che gli occhi pero' riconoscono come un movimento continuo. In altre parole abbiamo lo stesso sistema con cui vengono realizzati i filmati. E' possibile simultaneamente e indipendentemente animare uno o piu' sprite a varie velocita' e direzioni. Quando la visualizzazione di uno sprite viene disattivata dal comando sprite il suo movimento si ferma automaticamente. Una volta riattivato il comando il movimento tornera' automaticamente alle precedenti velocita' e direzioni. Vi diamo alcuni esempi del comando MOVSPR:

```
MOVSPR 1,160,100
```

Fissa l'angolo superiore sinistro dello sprite 1 al centro dello schermo ad alta risoluzione.

```
MOVSPR 1,+40,-60
```

Uso lo scarto relativo per muovere lo sprite uno

di 40 posizioni a destra e di 60 in alto.

MOVSPR 2,90 # 8

Anima lo sprite 2 ad un angolo di 90 gradi e ad una velocita' di 8.

NOTA: ricordiamo che e' possibile controllare la posizione e la velocita' degli sprites usando la funzione RSPPOS. Questa funzione ha 2 argomenti. Un numero di sprites e': un numero che richiede la posizione X la posizione Y o la velocita' dello sprite. La sintassi della funzione RSPPOS e' la seguente:

RSPPOS sprite, dati

L'argomento sprite identifica quale sprite si sta controllando. L'argomento dati specifica quale informazione debba essere restituita. Quando dati e' uguale a 0 viene ritornato la posizione corrente X dello sprite, quando dati e' uguale a 1 viene ritornata la posizione corrente Y dello sprite. E quando dati e' uguale a 2 viene ritornata la velocita' corrente dello sprite espressa in un numero da 0 a 15.
ATTENZIONE RSPPOS riporta sempre le coordinate di SCAL 0. Vediamo alcuni esempi della funzione RSPPOS:

RSPPOS (4,0)
Restituisce la posizione X dello sprite 4

RSPPOS (7,1)
Restituisce la posizione corrente sull'asse Y dello sprite 7.

100S=3:D=2:PRINT RSPPOS(S,D)

Stampa il valore attuale di velocita' dello

sprite 3.

GESTIONE DELLA COLLISIONE DEGLI SPRITES

I comandi Basic incrementati su Commodore 128 danno la possibilita' di rilevare quando qualcuno degli sprites in movimento entra in collisione con un altro sprite o comunque con una immagine sullo schermo. Normalmente questa immagine dovra' essere in BIT-MAP MODE sia standard che multicolor. Si possono rilevare le collisioni degli sprites usando il comando COLINT e determinare quali sprites sono entrati in collisione usando la funzione BUMP. Tutto cio' rende possibile creare visualizzazioni grafiche animate all'interno di programmi Basic.
Il comando COLINT individua 3 tipi di eventi: collisioni tra sprites, collisioni tra sprites ed immagini bit-map e attivazione della penna ottica o light pen. Quando si verifica uno di questi eventi il programma arresta la sua esecuzione e passa al primo numero di linea del sotto-programma di gestione delle collisioni.
In altre parole il programma Basic viene INTERROTTO e cede il controllo al sottoprogramma di collisione. Definiremo questi eventi come INTERRUPT DI COLLISIONE.
Una volta eseguito uno comando RETURN nel sottoprogramma di collisione il controllo viene restituito alla linea di programma successiva a quella che era stata interrotta. La sintassi del comando COLINT e' la seguente:

COLINT (N)[,n linea]

dove N puo' avere un valore di 0,1 o 2 e

specifica quale tipo di evento dovrebbe causare una interrupt di collisione. Se l'evento e' uguale a 0 vengono individuate le collisioni tra sprite e sprite; se evento e' uguale a 1 vengono individuate le collisioni degli sprite con la visualizzazione in bit-map. Mentre se l'evento e' uguale a 2 e' l'attivazione della penna ottica o LIGHT PEN a causare un interrupt. Il parametro numero di linea e' il primo numero di linea della subrutine presente nel programma Basic dove il programma stesso saltera' quando si verifichera' un interrupt di collisione del tipo definito da evento. Quando il parametro opzionale numero di linea viene dichiarato il rivelatore di collisione viene messo ON cioe' attivato per quel tipo di evento scelto. Quando il tipo di parametro numero di linea viene omesso il rivelatore di collisione viene messo su OFF cioe' disattivato per quel dato tipo di evento. Una collisione tra sprite e sprite interviene quando una parte di uno sprite che non sia dello stesso colore dello sfondo va a occupare la stessa posizione di una parte di un altro sprite anch'esso di colore diverso dallo sfondo.
Uno sprite non puo' causare un interrupt di collisione quando e' completamente fuori campo cioe' non e' visibile. C'e'invece collisione tra sprite e bit-map quando una parte di uno sprite che non sia dello stesso colore dello sfondo occupa la stessa posizione di una parte, ricordiamo sempre anch'essa di colore diverso da quella dello sfondo, di una immagine sullo schermo. E' da notare che gli sprites che sono stati disattivati tramite un opportuno comando SPRITE non causano interrupt di collisione.
Vediamo alcuni esempi del comando COLINT:

```
COLINT 0,400
```

Rileva la collisione tra sprite e sprite

COLINT 1,500

rileva la collisione fra sprite e immagine

COLINT :COLINT 1

disattiva le collisioni tra sprite e sprite e fra sprite e bit-map e quindi se e' attivo lascia solamente quella relativa all'impiego della penna ottica.

Negli esempi riportati il primo causa il trasferimento del controllo ad una subrutine che inizia alla linea 400 quando si verifica una collisione fra sprites. Allo stesso modo il secondo esempio provoca il trasferimento di controllo ad una subrutine alla linea 500 quando venga rilevata una collisione fra sprite e immagine. Nella linea 300 il rilevamento di ulteriori collisioni fra sprites viene disattivato durante la gestione dell'attuale collisione.
Si possono avere 1 o anche tutti i tipi di rilevamento di evento attivi allo stesso tempo ma solo una collisione alla volta puo' essere gestita. Percio' bisognerebbe sempre disattivare ulteriori rilevamenti di collisione come primo passo nei sottoprogrammi di gestione delle collisioni. Cio' impedisce che intervengano altri interrupt durante la gestione di quello corrente. Infine l'ultimo passo da eseguire nella subroutine di interrupt di collisione e' quello di riattivare il rilevatore di collisioni. Per individuare gli sprites che sono entrati in collisione si puo' utilizzare la funzione RBUMP. Questa funzione informa su quali sprites siano entrati in collisione con altri

sprites o su quali sprites siano entrati in collisione con la visualizzazione in modo bit-map. Non e' necessario che gli interrupt di collisione siano attivati per utilizzare RBUMP. La sintassi della funzione e':

RBUMP EVENTO

L'argomento RBUMP corrisponde direttamente al tipo di evento del comando COLINT. Se l'evento e' uguale a 0 la funzione RBUMP informa sulle collisioni fra sprite e sprite; se l'evento e' 1 RBUMP informa su quali sprites siano entrati in collisione con display bit-map. In ognuno dei casi precedenti RBUMMP restituisce un numero compreso tra 0 e 255.
La posizione di bit da 0 a 7 in un numero restituito da RBUMP corrispondono a numeri di sprite da 0 a 7. Quando un bit viene posto in ON (cioe' che ha un valore 1) lo sprite in questa posizione risulta che era stato coinvolto in una collisione. Qualora si verificassero contemporaneamente piu' collisioni, bisognera' usare anche la funzione RSPPOS di cui abbiamo parlato in precedenza per determinare quali sprites siano entrati in collisione e con quali oggetti. La funzione RBUMP legge i registri Hardware di collisione sprite nel chip di controllo video. Questi registri si azzerano automaticamente ogni qual volta vengano letti sia che si usi la funzione RBUMP che la funzione PEECK per leggerli direttamente. Percio' se si deve fare riferimento alle informazioni restituite da RBUMP piu' di una volta si dovra' assegnare il valore ad un nome di variabile.

MODO SPRITES DESIGNER

Il Commodore 128 e' dotato di un modo sprite designer che rende molto facile progettare e costruire degli sprites. Si puo' passare allo sprite designer sia in modo diretto sia da un programma. Infatti mentre si sta utilizzando il modo sprite designer l'esecuzione del programma basic e' sospesa e vengono abilitati divers controlli funzionali da tastiera per lo sprite designer. Per passare al modo sprite designer utilizzare il comando SPRDEF.
Questo comando non ha parametri.
Quando si passa in sprite designer lo schermo viene cancellato, cioe' avviene un CLEAR di schermo, e appare una larga area che servira' appunto per disegnare uno sprite nella parte sinistra dello schermo. Appena al di sotto di questa area appare il messaggio sprite number. Introdurre un numero da 1 a 8 che corrisponde allo sprite che si vuole definire o modificare.
E' bene fare attenzione che dopo il numero di sprite non si deve premere RETURN, perche' come si chiarira' meglio in seguito si uscira' da questo MODO.
A questo punto verra' visualizzato il contenuto dell' area di memoria relativa allo Sprite selezionato. Quest' area potra' essere vuota o contenere dei valori che saranno rappresentati in forma grafica. Si potra' eseguire anche il CLEAR di quell' area stessa.
Viene mostrata la dimensione reale dello sprite sul lato destro alto dello schermo.
Nell'angolo superiore sinistro dell'area riservata al disegno si potra' vedere un cursore che sara' diverso a seconda che si voglia disegnare sprites in alta risoluzione o in multicolor. Si possono ora utilizzare tasti di

controllo per muovere il cursore all'interno dell'area driservata al disegno. L'area riservata al disegno avra' le dimensioni di 24 X 21. Ogni posizione all' interno di questa griglia corrisponde a un pixel dello sprite che andremo a definire. Vediamo ora il sommario dei comandi che vengono abilitati quando ci troviamo in questo modo.

SOMMARIO DEI MODI DI DEFINIZIONE DEGLI SPRITES

TASTO CLR =cancella l'intera area di lavoro

TASTO M =attiva o disattiva lo sprite in multicolor

TASTO CTRL da ! a 8 =seleziona i colori per l'interno degli sprites da 1 a 8

TASTO COMMODORE PIU' TASTI DA 1 A 8 =seleziona il colore interno degli sprite per i colori da 1 a 16

TASTO 1 =attiva il pixel nel colore interno

TASTO 2 =attiva il pixel nei colori esterni

TASTO 3 =attiva un'area in multicolor 1

TASTO 4 =attiva un'area in multicolor 2

TASTO A =attiva o disattiva il movimento automatico dei cursori

TASTI CRSR = muovono il cursore oltre l'area di lavoro

RETURN =muove il cursore all'inizio della prossima linea cioe' esegue un ritorno a capo

TASTO HOME = muove il cursore nell'angolo sinistro in alto dell'area di lavoro

TASTO X = espande lo sprite orizzontalmente

TASTO Y =espande lo sprite verticalmente

RETURN + SHIFT =salva lo sprite dall'area di lavoro e restituisce come prompt il numero di sprite cioe'chiede un'altra volta il numero di sprite sul quale si vuole operare.

TASTO C =copia uno sprite su un altro

TASTO STOP =disattiva la visualizzazione dello sprite e riporta il prompt sprite number senza nessun cambiamento sullo sprite

L'ultimo tasto attivato e' il RETURN che pero' a differenza di quanto visto prima, quando ci chiede il numero dello sprite, usciremo dal modo di definizione sprite.

PROCEDURA PER LA CREAZIONE DI UNO SPRITE

Vediamo di illustrare una procedura che l'utente dovrebbe seguire per utilizzare correttamente e con il massimo risultato il modo di definizione sprite.

1)Eseguire la pulizia dell'area di lavoro premendo allo stesso tempo il tasto SHIFT e CLEAR HOME.

2)Se si desidera uno sprite in multicolore

premere il tasto M e verra' visualizzato un cursore addizionale insieme a quello normale. Il motivo per il quale appare un cursore doppio quando siamo in modo multicolore, e' perche' vengono impiegati due pixel.

3) Selezionate un colore per il vostro sprite. Per i colori tra 1 e 8 premere il tasto CONTROL e premere un tasto tra 1 e 8. Per selezionare i colori tra 9 e 16 premere il tasto COMMODOORE e premere sempre un tasto del tastierino numerico tra 1 e 8.

Ora siamo pronti per iniziare a creare il disegno dello sprite. I tasti numerati da 1 a 8 riempono lo sprite e ci danno il disegno.
Per uno sprite ad un solo colore utilizzare il tasto 2 per riempire l'area all'interno della griglia. Premere il tasto 1 per cancellare cio' che e' stato disegnato con il tasto 2. Se si desidera riempire una cella per volta premere il tasto A. In questo modo potremo muovere il cursore manualmente con i tasti normali di controllo cursore. Se si desidera che il cursore si muova automaticamente verso destra mentre si tiene premuto, non premere il tasto A poiche' questi fissa il movimento automatico del cursore.
E' da notare come abbiamo detto che ci troviamo di fronte a due aree di lavoro o meglio ad un'area di lavoro e ad un'area di immagine. Quando si opera sull'area di lavoro posta alla sinistra dello schermo sulla parte destra potremo vedere lo sprite come apparira' poi all'interno eventualmente di una animazione di una grafica di dimensioni maggiori e mano mano che si riempono le cellette all'interno della griglia potremo vedere l'accenzione o la visualizzazione dei corrispondenti pixel sulla parte destra dello schermo.

Quando ci troviamo in modo multicolore il tasto 3 riempe due cellette contemporaneamente all'interno della griglia con il colore selezionato per il multicolor 1 mentre il tasto 4 riempe sempre 2 cellette all'interno della griglia ma con il colore selezionato dal multicolor 2. Si possono cancellare delle cellette riempite all'interno dell'area di lavoro con il tasto 1. In modo multicolore il tasto 1 cancella due cellette per volta. L'operazione di cancellazione avviene togliendo il colore preimpostato dalla o dalle cellette nell'area di lavoro e rimettendo il colore di fondo dello schermo.
Ricordiamo che mentre si sta disegnando il nostro sprite ci possiamo muovere liberamente nell'area di lavoro quindi disegnare passondo sopra alle singole cellette in maniera trasparente utilizzando il RETURN per andare a capo, il tasto HOME per riportarsi in alto ed i cursori.

In qualsiasi momento si puo' espandere lo sprite sia in direzione verticale che orizzontale con i tasti Y e X. Per espandere verticalmente lo sprite premeremo il tasto Y, per epanderlo orizzontalmente il tasto X. Per ritornare alla normale dimensione dello sprite ripremere ancora una volta i tasti X o Y. Quando avete terminato di creare il disegno e siete soddisfatti di quello che avete fatto, e' necessariosalvare l'immagine. Questo avviene mediante la pressione del tasto SHIFT e RETURN. In questo modo il C128 salva i dati relativi allo sprite in una appropriata area che chiameremo area di immagazzinamento degli sprites. A questo punto verra' richiesto dal sistema lo sprite number. Se desideriamo uscire dal modo di definizione sprite, a questa domanda invece di rispondere con un numero da 1 a 8 per

selezionare un altro sprite premeremo semplicemente il RETURN. Se si desidera copiare uno sprite su un altro adoperare il tasto C.
Se non si desidera invece salvare il nostro sprite premere il tasto STOP. Il C128 disattiva lo sprite visualizzato e riformula ancora una volta la domanda "NUMERO DELLO SPRITE".
A parte il modo che abbiamo visto per uscire dal modo definizione sprite esistono altri due sistemi immediatamnete dopo aver salvato lo sprite con SHIFT RETURN e immediatamente dopo aver premuto il tasto STOP. Quando avete finito di creare lo sprite e siete usciti dal modo di definizione dello sprite la figura o meglio i dati relativi alla figura sono stati immagazzinati in una appropriata area che appunto si chiama area di immagazzinamento degli sprites all'interno della memoria del 128. Dopo l'uscita in qualsiasi modo sia stata fatta ritornerte in basic. Pero' a questo punto sara' anche necessario o per lo meno importante vedere lo sprite sullo schermo. Per attivarlo sara' necessario utilizzare il comando sprite che abbiamo gia' visto in precedenza. Per esmpio se avete creato lo sprite 2 nel modo di definizione sprite proviamo a digitare il seguente comando:

```
SPRITE 2,1,7,0,1,1,0
```

Lo visualizzeremo in questo modo lo coloreremo in blu e lo avremo espanso nelle direzioni X e Y. Ora utilizziamo il comando MOVSPR per muoverlo con un angolo di 45 gradi a velocita' 4. Utilizzeremo quindi:

```
MOVSPR 2,45 # 4
```

L'ultima funzione che ci rimane da vedere e'

quella del salvataggio del file su periferica perche' bisogna non dimenticare che una volta spento il computer tutto il nostro disegno andrebbe perso. Potremo impiegare quindi il comando BSAVE con questo formato:

```
BSAVE "nome del file",B0,P3584 TOP4096
```

Naturalmente per ricaricare il programma useremo il comando BLOAD con questi parametri

```
BLOAD "nome del file"
```

Vediamo ora delle particolarita' sul trattamento degli sprites.

LA MANIPOLAZIONE DEGLI SPRITES

Abbiamo appena imparato come creare, colorare, attivare o disattivare od animare uno sprite.
Puo' accadere pero' che si desideri un disegno che e' troppo grande o troppo dettagliato per essere immagazzinato in un singolo sprite. In questo caso bisogna congiungere due o piu' sprites in modo tale che il disegno possa essere piu' largo e piu' dettagliato che un singolo sprite. Dopo aver unito gli sprites ognuna di queste aree di disegno puo' muoversi indipendentemente l'uno dall'altro. Cio' da' un maggior controllo su tutta l'animazione che un singolo sprite.
Il concetto fondamentale di unire due Sprites e dopo di farli muovere insieme consiste nel procedimento del disegno prima, usando le tecniche grafiche, cioe' i comandi DRAW, BOX ecc., visti in precedenza.
Il passo successivo sara' quello della

memorizzazione in due o piu' aree differenti usando il comando SSHAPE e quindi di attivarli dopo averli posizionati.
La posizione degli sprites in congiunzione fra loro deve essere fatta in modo tale che l' inizio del secondo Sprite sia al successivo PIXEL dalla fine del precedente.
Proviamo per esempio a disegnare una figura poniamo di 48 per 42 pixel. In questo caso posizioneremo il primo sprite e di seguito, a 24 pixel il secondo con i seguenti comandi, nell' area 1 e 2 degli sprites. Osservare nei comandi che riportiamo la posizione delle coordinate X e Y.

```
10 MOVSPR 1,10,10
20 MOVSPR 2,34,10
```

Infine gestire un piccolo programma che li muova contemporaneamente.
In definitiva il segreto consiste nel sincronismo.

IMMAGAZZINAMENTO DEGLI SPRITES

Il C128 ha due comandi, BLOAD e BSAVE che consentono di manipolare dati relativi agli sprites in maniera abbastanza facile, anche se le seguenti spiegazioni presuppongono da parte dell'utente almeno una certa conoscenza del linguaggio macchina e quindi della gestione della memoria e dei codici dei files come codici oggetto.
Il B nelle parole BLOAD e BSAVE sta per binario. Infatti questi comandi consentono di salvare e caricare files binari dall'unita' a dischi. (Per quanto riguarda la gestione dei comandi per

l'unita' a dischi si rimanda all'apposito capitolo sulla gestione delle periferiche presente in questa guida).
Il file binario puo' essere composto sia da un programma in linguaggio macchina o da una subroutine in linguaggio macchina come parte di un programma principale in basic oppure da un insieme di dati compresi fra due indirizzi di memoria.
Quando si adopera questo tipo di comando SAVE avremo memorizzato su disco un file che verra' considerato dal sistema come file binario.
Il file binario e' piu' facile da manipolare che un file di codice oggetto perche' il file binario puo' essere caricato senza nessuna routine preparatoria.
Un file codice oggetto deve essere caricato con un LOADER e quindi mandato in esecuzione tramite un comando di sistema tipo SYS.
Bisogna fare attenzione perche' quando si carica un file binario si deve seguire uno di questi due sistemi:

```
LOAD"nome del file",8,1
```

Oppure

```
BLOAD"nome del file"B0,P inizio
```

dove l'inizio sara' la locazione di memoria decimale 3584 se si sta caricando in file relativo agli sprites.
Con il primo metodo visto bisogna specificare ",1" al termine del comando di caricamento perche' in caso contrario il computer lo considerera' un programma basic e lo carichera' all'inizio dell'area basic. Infatti il ",1" comunica al sistema che si sta caricando un file binario e che quindi questi deve essere immesso a partire dalla stessa locazione di memoria

dalla quale era stato salvato.
Vediamo il motivo di questi numeri e del modo di comportarsi del sistema.
Il Commodore 128 ha dedicato all'immagazzinamento degli sprites una determinata zona di memoria che va dall'indirizzo decimale 3584 ($0E00) fino all'indirizzo decimale 4095 ($0FFF) per un totale di 512 bytes.
Quest'area di memoria e' prevista per l'immagazzinamento di 8 sprites ognuno dei quali come sappiamo e' composto da 24 pixels di larghezza per 21 pixels di altezza, per un totale quindi di 504 pixels. Poiche' ogni pixel corrisponde a un bit che e' l'ottava parte di un byte possiamo affermare che per ogni sprite sono necessari 63 bytes di memoria. In effetti il C128 usa un byte in piu' per cui l'area di memoria per ogni sprite sara' di 64 bytes. Da questi dati (64 bytes x 8 sprites = 512 bytes) si deduce che l'area di memoria necessaria e' di 512.

Come abbiamo gia' accennato l' intera area dove risiedono i valori di tutti e 8 gli Sprite va da 3584 decimale a 4095 ($0E00 a $0FFF). Vediamone una tabella:

INDIRIZZI DI MEMORIA PER GLI SPRITES

HEX	DEC	N. SPRITE
0E00	3584	1
0E40	3648	2
0E80	3712	3
0EC0	3776	4
0F00	3840	5
0F40	3904	6
0F80	3968	7
0FC0	4032	8

Il termine come detto sara' dato dall' ultimo indirizzo di inizio piu' 63 e percio' sara' a 4095 ($0FFF).
Vediamo ora il comando per il salvataggio degli sprites non appena si esce dal modo di definizione:

```
BSAVE "nome",B0,P3584 TO P4096
```

Il nome e' quello scelto dal programmatore e potra' essere uno qualsiasi. Il B0 specifica che stiamo operando sul Banco di memoria 0. I valori relativi agli indirizzi ci danno i dati di partenza e quelli di arrivo, in questo caso la zona di memoria che salveremo e' quella relativa agli Sprites. Non dimentichiamo infatti che i comandi BLOAD e BSAVE possono essere impiegati ANCHE per salvare zone di memoria qualsiasi.
L' ultima osservazione da fare e' che nel comando BSAVE va sommato 1 alla locazione finale di memoria che si vuole salvare. Infatti si salva fino a 4096 e non fino a 4095.
Ricordiamo invece che nel comando BLOAD i parametri relativi al banco su cui si vuole ricaricare il programma e l' indirizzo di partenza sono opzionali. Solo se si desidera cambiare il banco di caricamento oppure l' indirizzo di memoria questi vanno dati.

QUESTA PAGINA E' STATA LASCIATA INTENZIONALMENTE BIANCA

CAPITOLO QUARTO

PROGRAMMAZIONE DEI SUONI E DELLA MUSICA CON IL COMMODORE 128

Il Commodore 128 e' dotato di uno dei piu' sofisticati sintetizzatori elettronici musicali disponibili su HOME COMPUTER.
Questo sintetizzatore chiamato **SOUND INTERFACE DEVICE (SID)** e' un circuito integrato dedicato solo ed esclusivamente alla generazione di suoni e di musica. Il SID e' completo di tre voci totalmente indirizzabili, un generatore ADSR (Attack Decay Sustain and Release) filtratura, modulazione e rumore bianco.
Ognuna di queste tre voci e' indipendente e puo' essere programmata simultaneamente. Inoltre ognuna di queste voci puo' essere programmata in uno dei quattro tipi di suoni chiamati forme d'onda.
I parametri del ADSR definiscono invece la qualita' del suono. Tutte queste grandi capacita' sono disponibili attraverso poche e semplici istruzioni del Basic 7.0 implementato sul modo 128.
Questo sta a significare che si possono comporre brani musicali e suoni anche complessi utilizzando tecniche di programmazione relativamente semplici.
Questa parte della GUIDA e' stata concepita per agevolare la comprensione di tutte le capacita' del circuito 6581 appunto il SID.
Verranno quindi spiegate in parte le teorie musicali sia gli aspetti pratici che si incontrano nella conversione di tale teoria in

pratica. Non sara' necessario quindi essere un programmatore o un musicista esperto per raggiungere risultati validi con questo potente sintetizzatore musicale.
I comandi Basic per l'incrementazione dei suoni ognuno dei quali avra' dei singoli parametri sono i seguenti:

SOUND
ENVELOPE
VOL
TEMPO
PLAY
FILTER

Vedremo di spiegare questi comandi uno per volta e poi di riunirli tutti insieme in modo tale da avere un unica gestione delle qualita' sonore.
Il comando che tratteremo piu' approfonditamente anche perche' in effetti e' il piu' importante e' il comando SOUND che incominciamo ad esaminare in tutti i possibili parametri che lo compongono.

IL COMANDO SOUND

Il comando SUOND e' stato messo a punto con lo scopo primario di programmare facilmente e rapidamente degli effetti sonori. Dispone di una serie di parametri abbastanza imponente una parte dei quali sono opzionali e come abbiamo detto da' un notevole aiuto alla programmazione sonora. Il formato del comando SOUND e' il seguente:

```
SOUND VC, FREQ,DUR(,DIR(,MIN(,SV(,WF(,PW)))))
```

i cui parametri sono:

VC - seleziona la voce 1, 2 o 3
FREQ - fissa il livello di frequenza sonoro con un valore che puo' andare da 0 a 65535
DUR - fissa la durata del suono in sessantesimi di secondo
DIR - fissa la direzione in cui il suono viene incrementato o decrementato, per cui avremo:
0 = incremento della frequenza;
1 = decremento della frequenza;
2 = oscillazione della frequenza in alto e in basso.
MIN - seleziona il minimo della frequenza con un valore che va da 0 a 65535 se la direzione (cioe' il parametro visto precedentemente) e' specificato
SV - sceglie il valore di passo per la direzione. Il valore di questo parametro puo' essere da 0 a 32767
WF - seleziona la forma d'onda per cui sara':
0 = triangolo
1 = dente di sega
2 = a impulso variabile
3 = rumore bianco.
PW - seleziona l'ampiezza d'onda

Ricordiamo che i parametri (DIR, MIN, SV,WF e PW) sono opzionali.

Il primo parametro VC serve a selezionare una delle tre voci che abbiamo a disposizione. Il secondo parametro FREQ determina la frequenza del suono e come abbiamo detto sara' possibile assegnargli un valore tra 0 e 65535. Il terzo fissa la durata cioe' il tempo in cui il suono deve essere suonato. La durata viene misurata in 60mi di secondo per cui se si vuole far durare un suono per un secondo si sceglie una durata a

60. Per suonare per 2 secondi richiederemo una durata di 120, per suonare 10 secondi una durata di 600 e cosi' via.

LA FREQUENZA E LE ONDE SONORE

Senza voler fare un esame estremamente approfondito ricordiamo che il suono e' generato in forma di onde dal movimento dell'aria. Se si getta un sasso in uno stagno si possono osservare le onde che si allontanano a raggera dal punto dell'impatto.
Allo stesso modo quando onde simili si creano nell'aria siamo in grado di udirle. Se si misurano due successivi picchi d'onda si trova il numero di secondi per cicli dell'onda. Il reciproco di questo numero (1/n) da il numero di cicli per secondo. Questa quantita' e' conosciuta anche come FREQUENZA. L'acuto o il profondo di un suono (la nota) sono determinati dalla frequenza delle onde sonore prodotte. Il quarto parametro DIR seleziona la direzione in cui la frequenza sonora e' incrementata o decrementata. Il quinto MIN fissa la minima frequenza in cui deve incominciare lo sweep. Il sesto e' il passo dello sweep. Questo parametro e' similare al passo nel comando FOR NEXT. Se i valori DIR, MIN e SV sono dichiarati nel comando SOUND allora il suono prodotto incomincia a livello dichiarato dal parametro FREQ. Quindi il sintetizzatore passa in mezzo e suona ogni livello dell'intero intervallo di valori di frequenza incominciando dalla frequenza minima determinata dal parametro MIN.
Lo sweep e' incrementato o decrementato dal valore di passo (SV) in funzione della direzione specificata dal parametro DIR e la frequenza e'

suonata ad un nuovo livello. Il settimo parametro WF di questo comando seleziona la forma d'onda o waveform il cui significato sara' visto in seguito. L'ultimo parametro nel comando suono determina l'ampiezza d'impulso della forma d'onda se questa e' stata selezionata.
Vediamo di scrivere ora qualche programma o meglio di fare delle prove:

```
20 SOUND 1, 4096, 60
```

Facciamo girare questo programma con RUN. Udremo ora un beep acuto. Che cosa abbiamo fatto? Con questa linea abbiamo messo in funzione la voce numero 1 alla frequenza di 4096 per la durata di un secondo. Proviamo a cambiare la frequenza di questo programma:

```
30 SOUND 1, 8000, 60
```

Dando il RUN 30 possiamo notare che questo nuovo comando fa eseguire un suono ad un livello tonale piu' alto. Cio' mostra la diretta relazione tra la frequenza tra differenti frequenze.
In altre parole se si incrementa la frequenza il SID incrementa il picco del tono. Proviamo ora con un nuovo comando:

```
40 SOUND 1, 0, 60
```

Con la linea 40 abbiamo dato alla frequenza il valore minimo che e' 0 e che e' la piu' bassa frequenza disponibile anzi e' cosi' bassa da essere quasi non udibile. Ricordiamo che un valore di frequenza di 65535 e' la frequenza piu' alta possibile.

Vediamo ora di esaminare sempre con un esempio la gamma delle frequenze disponibili.

Utilizziamo un ciclo di FOR NEXT per questo.

```
50 FOR I = 1 TO 65535 STEP 1000
60 SOUND 1,I,1
70 NEXT
```

Questo piccolo programma suona la forma d'onda a variabile di impulsi nell'intervallo di frequenze da 1 a 65535 con un incremento di 100 dalla piu' bassa alla piu' alta frequenza possibile. Ricordiamo che se non si specifica la forma d'onda il computer seleziona per the forth il valore 2 della forma d'onda che e' appunto la forma d'onde a variabile d'impulsi.

Cambiamo ora la forma d'onda con la seguente linea di programma cioe' cambiamo in altre parole la linea 60 del programma precedentemente scritto:

```
60 SOUND,1,I,1,0,0,0,0,0
```

Ora il programma fa suonare sempre la voce 1 utilizzando la forma d'onda a triangolo nell'intervallo di frequenza da 1 a 65535 sempre con l'incremento di 100.

Proviamo a scrivere una nuova linea di programma da aggiungere al piccolo programmino precedente.

```
200 SOUND 1,49152,240,1,0,100,1,0
```

Questa linea di programma fa iniziare la frequenza a 49152 e decrementa lo sweep di 100 in 100 nella direzione basso fino a quando non trova il minimo sweep a 0. La voce 1 utilizzando la forma d'onda a dente di sega consente l'emissione di ogni sound per 4 secondi. Udremo normalmente un suono simile allo scoppio di una bomba.

Proviamo ora a cambiare alcuni dei parametri nella linea 100. Per esempio cambiamo la direzione dello sweep in 2 (oscillazione) cambiamo il minimo della frequenza di sweep a 32768, incrementiamo il passo a 3000. Il nuovo comando SOUND sara' scritto quindi cosi':

```
210 SOUND 1,49152,240,2,32768,3000,1
```

Avremo un suono simile alla sirena della polizia. Fino a questo momento abbiamo programmato utilizzando una sola voce. Si possono produrre effetti sonori veramente molto interessanti con il comando SOUND utilizzando fino a 3 voci. Provate a creare un programma che utilizzi insieme tutte e 3 le voci.
Ecco ora un esempio di programma che dovrebbe aiutarvi a comprendere meglio l'utilizzo del sintetizzatore SID presente sul 128. Il programma quando gira eseguira' una richiesta per ogni parametro del comando SOUND.

```
10 PRINT:PRINT:PRINT:PRINT"            SOUND
PLAYER":PRINT:PRINT:PRINT
20 PRINT"IMMETTI I PARAMETRI PER
SUONARE":PRINT:PRINT
30 INPUT "VOCE (1-3)":V
40 INPUT "FREQUENZA (0-65535)":F
50 INPUT "DURATA (0-32767)":D:PRINT
60 INPUT"VUOI DEI PARAMETRI OPZIONALI?
S/N":B$:PRINT
70 IF B$="N" THEN 130
80 INPUT "DIREZIONE SWEEP
0=SU,1=BASSO,2=OSCILL":DIR
90 INPUT "MIN. FREQUENZA SWEEP (0-65535)":M
100 INPUT "PASSO DI SWEEP (0-32767)":S
110 INPUT "ONDA ":W
120 IF W=2 THEN INPUT "AMP. D' IMPULSO
(0-4095)":P
```

```
130 SOUND V, F, D, DIR, M, S, W, P
140 INPUT"DOBBIAMO RIPETERE ? S/N":A$
150 IF A$="s"THEN 130
160 GOTO10
```

NOTA

Programma dal manuale CBM.

Vediamo una spiegazione sintetica ma chiara del programma.
Le linee da 10 a 20 visualizzano un messaggio introduttivo sullo schermo. Le linee da 30 a 50 richiedono la voce, la frequenza e i parametri di durata.
La linea 60 chiede se si vogliono aggiungere parametri addizionali. Se non si vogliono dare questi parametri addizionali premere il tasto "N" e il programma saltera' direttamente alla linea 120 ed eseguira' il suono secondo l'impostazione data in precedenza. Se invece si voglio specificare i parAmetri addizionali del comando SOUND premere il tasto "S" e il programma andra' a eseguire la linea 80. Le linee da 80 a 110 specificano la direzione di sweep, la frequenza minima di sweep, il valore del passo di sweep e la forma d'onda.
La linea 120 esegue l'ingresso dei valori relativi alla forma d'onda solo se la forma d'onda e' stata selezionata.La linea 130 esegue il comando SOUND relativamente ai parametri selezionati. Alla linea 140 si chiede se si voglia riudire ancora il suono di cui sono stati passati i parametri in precedenza. Questa funzione viene eseguita se si preme il tasto "S" mentre se si preme il tasto "N" il programma ripartira' dall'inizio.

SUONI CASUALI

Vediamo ora sempre a livello dimostrativo un programma che genera suoni casuali utilizzando la funzione RND. Come al sollito si consiglia di scrivere questo programma e di salvarlo prima di farlo girare. La funzione specifica di questo programma e' di illustrare quante centinaia di combinazioni di suoni si possono produrre utilizzando i parametri del comando SOUND:

```
10 PRINT"VC  FREQ  DIR  MIN  SV   WF   PW "
20 PRINT"
30 V=INT(RND(1)*3)+1
40 F=INT(RND(1)*65535)
50 D=INT(RND(1)*32767)
60 DIR=INT(RND(1)*3)
70 M=INT(RND(1)*65535)
80 S=INT(RND(1)*32767)
90 W=INT(RND(1)*4
100 P=INT(RND(1)*4095
110 PRINTV: F;DIR;M;S;W;P:PRINT:PRINT
120 SOUND V, F, D, DIR, M, S, W, P
130 SLEEP 4
140 SOUND V, 0, 0, DIR, 0, 0, W, P
150 GOTO10
```

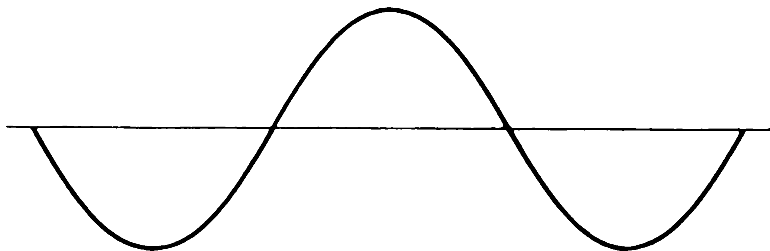
Vediamone una breve spiegazione:
Le linee 10 e 20 servono per l'intestazione della pagina. Le linee dalla 30 alla 100 calcolano ogni parametro del comando suono entro un determinto intervallo. Per esempio la linea 30 calcola il numero di voci da selezionare come segue:

```
30 V=INT(RND(1)*3)+1:REM VOCE
```

In questa linea si puo' notare che il parametro RND del basic C128 genera un numero compreso fra 0 e 3. Per la generazione dei numeri casuali vedi quanto detto sia nel capitolo dei comandi.
La linea 110 serve a visualizzare i valori che assumono i parametri in funzione del generatore di numeri casuali.
La linea 120 fa eseguire al sistema il comando SOUND con i parametri appena specificati. La linea 130 genera un ritardo di 4 secondi. Lalinea 140 disabilita il comando SOUND dopo il ritardo di 4 secondi. Prima di passare ai comandi successivi che il Basic del C128 mette a disposizione vediamo alcune caratteristiche sul suono.

LE CARATTERISTICHE DEL SUONO

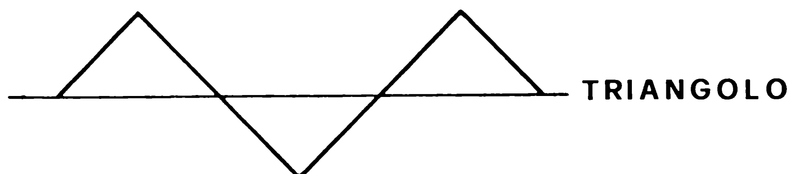
Riprendendo il discorso fatto a proposito delle onde che generano il suono, possiamo dire che la qualita' tonale di un suono si chiama timbro.
Il timbro di un suono e' determinto essenzialmente dalla sua forma d'onda. Se si ricorda l'esempio del sasso gettato nell'acqua si ricordera' anche che le onde si propagano uniformemente sullo stagno. Queste onde assomigliano abbastanza alla prima onda sonora di cui parliamo: l'onda sinusoinale che e' illustrata nella figura seguente.
Per rendere un po' piu' evidente l'argomento di cui stiamo parlando, illustreremo le forme d'onda con dei disegni e da questi disegni si potra' anche capire per quale motivo viene assegnato ad ogni onda un determinato nome.

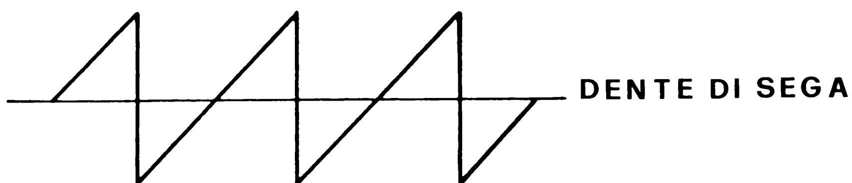


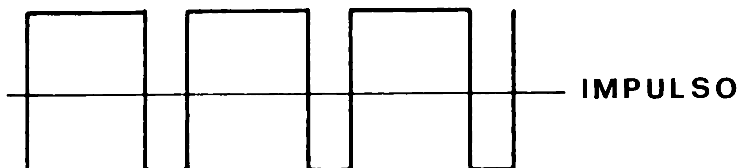
Una nota che viene suonata e' formata da un'onda sinusoidale oscillante alla frequenza fondamentale e alle armoniche di quell'onda. La frequenza fondamentale definisce in maniera completa la tonalita' della nota. Le armoniche sono onde sinusoidali la cui frequenza e' un multiplo intero della frequenza fondamentale. Un'onda sonora e' composta dalla frequenza fondamentale e da tutte le armoniche richieste per formare quel suono.
La teoria musicale parte dal presupposto che l'armonica numero 1 sia la frequenza fondamentale, che la seconda armonica abbia una frequenza doppia di quella fondamentale, la terza una frequenza tripla ecc... La quantita' di ogni armonica presente in una nota da' il timbro della nota stessa. Uno strumento acustico come una chitarra, un violino, un pianoforte hanno una struttura armonica molto complessa per il fatto che tale struttura puo' variare a seconda di come viene variata una singola nota.
Le forme d'onda disponibili sul sintetizzatore del C128 come abbiamo detto sono:

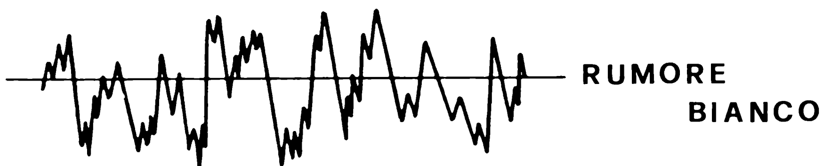
DENTE DI SEGA
TRIANGOLO
A IMPULSO VARIABILE
RUMORE BIANCO

Vediamole un po' piu' approfonditamente.
Un'onda triangolare possiede soltanto armoniche casuali. La quantita' di ogni armonica presente e' proporzionale al reciproco del quadrato del numero di armonica. In altre parole l'armonica numero 3 e' 1/9 piu' dolce dell'armonica numero 1 in quanto il quadrato di 3 e' 9 e il suo reciproco 1/9. Guardando i disegni si puo' osservare che c'e' una certa somiglianza nella forma di un'onda triangolare rispetto ad un'onda sinusoidale oscillante alla frequenza fondamentale. Un'onda a dente di sega contiene tutte le armoniche. La quantita' di ogni armonica presente e' proporzionale al reciproco del numero di armoniche. Ad esempio l'armonica numero 2 e' profonda 1/2 rispetto all'armonica n. 1. L'onda rettangolare contiene armoniche casuali in proporzione al reciproco del numero di armoniche. Onde rettangolari diverse hanno un diverso contenuto armonico. Cambiando l'ampiezza dell'impulso viene modificata grandemente l'ampiezza del suono di un onda rettangolare. Scegliendo accuratamente la forma d'onda usata si puo' dare inizio ad una struttura armonica che assomiglia in qualche modo al suono che si desidera riprodurre. Per la rifinitura di tale suono dovremo aggiungere un'altra caratteristica delle q ualita' del suono disponibile tramite il SID chiamata filtratura della quale parleremo piu' avanti.


TRIANGOLO

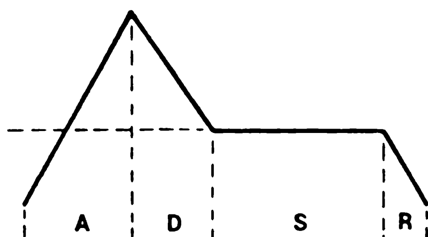

DENTE DI SEGA


IMPULSO


RUMORE
BIANCO

IL GENERATORE DI INVILUPPO

Il volume di un tono musicale cambia dal momento in cui viene percepito via via fino alla sua scomparsa quando non puo' piu' essere udito. Quando la nota viene suonata per la prima volta il suo volume sale da 0 al suo volume di picco, cioe' al suo volume piu' alto.
Il passo in cui cio' si verifica si chiama ATTACCO. Successivamente la nota scende di volume dal valore di picco ad un valore medio.
Questo passo prende il nome di decadimento mentre il livello medio raggiunto si chiama LIVELLO DI SOSTEGNO.
Quando alla fine al nota cessa di suonare, il volume non passa immediatamente dal livello di sostegno al livello 0 ma attraversa una serie di valori che generano appunto un livello. Questo livello si chiama livello di RILASCIO. Il disegno seguente mostra queste 4 fasi.



Ognuno dei 4 livelli sopradescritti conferisce ad ogni nota certe qualita' e dimensioni. I 4 livelli si chiamano parametri e vengono indicati tutti insieme con le rispettive iniziali in inglese: Attack, Decay, Sustain e Release abbreviati in ADSR. Il generatore di inviluppo controlla i parametri dell'ADSR.
Nel Basic del 128 esiste proprio un comando ENVELOP, che vedremo subito dopo e che consente di cambiare ogni parametro ADSR assegnandogli 16 differenti valori. Cio' da una enorme flessibilita' al generatore di inviluppo e consente dei risultati notevoli nella generazione del suono. Il formato del comando ENVELOP e' il seguente:

```
ENVELOPE e(,a(,d(,s(,r(,wf(,Pw))))))
```

I parametri di un comando hanno i seguenti significati:

e - e' il numero di inviluppo da 0 a 9
a - e' l'attack rate da 0 a 15
d - e' il decay rate da 0 a 15
s - e' il livello di sostegno da 0 a 15
r - e' il livello di rilascio da 0 a 15
wf- e' la forma d'onda che abbiamo gia' spiegato in precedenza
pw- e' l'ampiezza d'impulso che potra' andare da 0 a 4095.

Il C128 ha 10 predefiniti inviluppi per 10 differenti stumenti musicali. Utilizzando questi numeri predefiniti non sara' piu' necessario specificare i parametri ADSR, la forma d'onda e l'ampiezza d'impulso perche' questi vengono scelti automaticamente dal Commodore 128. Vediamo la tabella con i differenti tipi di inviluppo.

PARAMETRI DI DEFAULT PER IL COMANDO ENVELOPE

0	Pianoforte	0	9	0	0	2	1536
1	Fisarmonica	12	0	12	0	1	
2	Calliope	0	0	25	0	0	
3	Tamburo	0	5	5	0	3	
4	Flauto	9	4	4	0	0	
5	Chitarra	0	9	2	1	1	
6	Clavicembalo	0	9	0	0	2	512
7	Organo	0	9	9	0	2	2048
8	Tromba	8	9	4	1	2	512
9	Xilofono	0	9	0	0	0	

Nella prima colonna abbiamo il numero di inviluppo, nella seconda lo strumento. Nelle altre rispettivamente, Attacco, Decadimento, Sostegno, Rilascio, forma d' onda e ampiezza.

IL VOLUME

Il passo successivo nella programmazione di note musicali e' di fissare il volume del SID con un comando molto semplice.
Vediamo 20 VOL 10

Con questo comando si puo' fissare il volume a cui sara' prodotta la nostra musica o comunque i nostri effetti sonori con un valore compreso in un intervallo fra 0 e 15 dove 15 e' il massimo e a 0 non abbiamo nessun volume.

TEMPO

Ora fisseremo il TEMPO ovvero la velocita' della nostra esecuzione musicale. Il comando tempo viene usato per determinare la velocita' alla quale la musica verra' suonata. La sintassi del comando tempo e':

TEMPO n

dove il parametro "n" e' uguale alla velocita' e controlla la durata relativa delle note mentre vengono suonate. Il valore della velocita' puo' variare da 0 a 255. Al valore 0 la nota suonera' indefinitivamente. Con l'aumentare del valore velocita' la durata della nota diminuisce. Il valore di default e' 8.

NOTA

Si puo' calcolare la durata effettiva dell' intera nota attraverso la formula in secondi:

DURATA: 19.22/n

dove "n" e' la velocita'.

IL COMANDO PLAY

Con il 128 e' possibile comporre musica utilizzando caratteri stringa. Si potra' eseguire la musica introducendo questi caratteri e includendoli all'interno di parentesi nel comando PLAY. Questo comando funziona allo stesso modo che abbiamo visto per il PRINT. Il formato generale di questo comando infatti e':

PLAY"stringa del sintetizzatore"

Il numero totale di caratteri incluso le note musicali e i caratteri di controllo del sintetizzatore che possono essere inseriti entro un comando PLAY possono essere un massimo di 255.
Tuttavia poiche' questo eccede il numero massimo di caratteri (160) consentito per una singola linea di programma del basic, dovrete concatenare (aggiungendole col segno + ad esempio A$+B$) almeno due stringhe per raggiungere questa lunghezza. Sarebbe meglio pero' evitare di concatenare stringhe e non superare i 160 caratteri. Le note vengono specificate con lettere dalla A alla G (dal LA al SOL) la durata viene indicata delle lettere W (whole=intero), H (half=meta') Q (quarter=un quarto), I (Eighth=un ottavo) e con S (sixteenth=un sedicesimo). Ogni nota che segue una delle lettere di durata viene suonata alla stessa lunghezza fino a che la durata stessa non viene cambiata. La lettera R dichiara una pausa nella durata della nota.
Le note che vengono precedute dal segno # vengono suonate con il diesis e le note precedute dal segno $ vengono suonate come bemolle. Le note precedute da un . vengono suonate come note puntate una volta e mezzo della durata della note normali. Vediamo uno schema che riassume questi elementi musicali:

ELEMENTO	DESCRIZIONE
A,B,C,D,E,F,G	Note
#	Diesis
$	Bemolle
.	Puntata
W	Intero

H	Meta'
Q	Quarto
I	Ottavo
S	Sedicesimo
R	Pausa

Inoltre si puo'comunicare al computer di attendere fino a quando le voci selezionate attualmente per suonare arrivino, cioe' non giungano alla fine del loro lavoro includendo il seguente comando "N". Allo stesso modo le lettere vengono usate per definire certi valori di controllo del SID. Si possono inserire il controllo del volume principale usando la lettera U seguita da un numero da 0 fino a 9. Una o piu' delle 3 voci del SID possono essere attivate allo stesso tempo. La lettera V seguita da un numero da 0 a 2 scegliera' quali voci dovranno eseguire la musica. Il filtro del SID viene messo su un'onda X1 e su ESPACE 0. Essendoci un solo filtro le impostazioni influenzano tutte le voci attivate. Un'ottava particolare per una nota viene scelta dalla lettera O seguita da un numero da 0 a 6. Vediamo comunque una tabella descrittiva deicaratteri di controllo che possono essere inseriti all'interno del comando ENVELOPE con i vari valori e significati:

CARATTERE CONTROLLO	DESCRIZIONE	INTERV	DEFAULT
Vn	Voce	1-3	1
On	Ottava	0-6	4
Tn	Inviluppo	0-9	0
Un	Volume	0-15	9
Xn	Filtro	0=off, 1=on	0

Sebbene il SID possa processare questi caratteri di controllo in un ordine qualsiasi per ottenere un miglior risultato si consiglia di rispettare l'ordine con il quale sono stati descritti nella tabella precedente.
Se tutte le prescrizioni e sopratutto i parametri NON verranno osservati, potremmo avere delle brutte sorprese, cioe' aver programmato un tipo di suono e ottenerne un altro.

IL FILTRO

Il contenuto sonoro di una forma d'onda puo' essere modificato usando un filtro. Il circuito SID e' equipaggiato con tre tipi di filtri che possono essere usati singolarmente o in combinazione. Possiamo quindi utilizzare un filtro passa alto che lascia passare tutte le frequenze maggiori o uguali a quelle di taglio mentre attenua le frequenze al di sotto di quelle di taglio. Vi e' un altro filtro che e' il filtro passa basso che lascia passare le frequenze al di sotto di quelle di taglio ed attenua quelle al di sopra. Infine avremo a disposizione il filtro passa banda che lascia passare una banda di frequenze ristrette intorno alla frequenza di taglio attenuando tutte le altre .Ricordiamo che la frequenza di taglio e' il punto di riferimento del filtro ed e'........

I filtri passa alto e passa basso possono essere combinati per formare un filtro di rigetto del taglio che attenua la frequenza di taglio lasciando passare tutte le altre. Il comando

FILTER viene usato quindi per variare dinamicamente altre qualita' tonali del suono prodotto. Si potra' farlo regolando un filtro d'onda per sopprimere le gamme di frequenze scelte. Si potra' anche specificare l'effetto di risonanza che enfatizza le note con frequenze vicine a quelle di taglio del filtro. La sintassi del comando FILTER e':

FILTER cf, lp, bp, hp, res

in cui:

cf -e' la frequenza di taglio; valore che puo' andare da 0 a 2047
lp -e' il filtro passa basso che puo' essere a 0 o 1 cioe' attivo o inattivo
bp -e' il filtro passa banda
hp -e' il filtro passa alto
res-e' la risonanza

Normalmente i parametri passa basso, passa alto e passa banda vengono usati insieme per determinare quali parti dello spettro audio debbano passare inalterate all'uscita del SID e quali parti devono essere espresse dal filtro. Come abbiamo visto ognuno di questi parametri puo' avere un valore da 0=soppressione a 1=passaggio. Si potranno impostare uno o piu' di questi parametri su uno dei due valori.
Il parametro RES per la risonanza puo' variare da 0 a 15 e determina la risonanza per esempio quanto venga enfatizzato l'effetto massimo dei suoni in prossimita' delle frequenze di taglio.
Vediamo alcuni esempi:

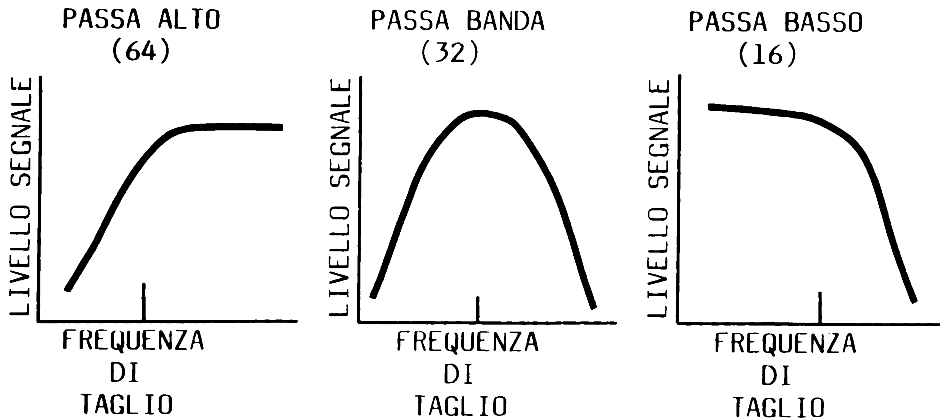
FILTER 1200, 1, 0, 0, 10

In questo esempio fissiamo la frequenza di taglio a 1200, attiviamo il filtro passa basso

disabilitiamo il filtro passa alto e passa banda e assegnamo un livello di risonanza pari 10. Facciamo girare con questo tipo di filtraggio un programma sonoro poi premendo RUN STOP-RESTORE resettiamo il nostro chip. Cambiamo la precedente linea con questo valore:

```
FILTER 1200, 0, 1, 0, 10
```

Questa nuova linea disabilita il filtro passa basso e abilita il filtro passa banda. Noterete nel far suonare sempre un programma sonoro la differenza.
Le figure sotto mostrano le forme d' onda prodotte dai vari effetti di filtraggio.



CAPITOLO QUINTO

LE PERIFERICHE

Premettiamo che quanto segue e' in gran parte tratto dal manuale EVM LE PERIFERICHE COMMODORE al quale rimandiamo per gli approfondimenti.
E' normalmente molto difficile che un computer possa essere utilizzato come sola unita' centrale. In questo caso infatti ci si dovrebbe limitare ad utilizzare un mezzo sicuramente meno potente, ma altrettanto sicuramente piu' facile da usare, come ad esempio una calcolatrice delle quali ne esistono versioni sempre piu' sofisticate e con grandi capacita'.
Esiste pur sempre, utilizzando un computer piccolo o grande che sia, il problema di immagazzinare programmi scritti dall' utente stesso oppure la necessita' di caricare programmi acquistati. Percio' un computer diventa realmente un SISTEMA DI ELABORAZIONE DATI solo quando dispone come minimo di un' unita' di massa o di memoria esterna.
Un sistema di computer e' quindi composto da qualcosa in piu' di una tastiera, di uno schermo e della stessa unita' centrale, cuore del computer. Per avere un programma a disposizione tutte le volte che si vuole senza doverlo reinserire nella memoria della unita' centrale con una lunga e noiosa serie di operazioni durante la quale e' oltretutto facile sbagliarsi e' necessario disporre di una unita' di registrazione che potra' essere una unita' a cassetta o un floppy disk.
Prendiamo per ipotesi un programma per la gestione di indirizzi o MAIL PROGRAM. Questo programma, per non riscriverlo tutte le volte,

dovra' essere immagazzinato in una cassetta o in un dischetto. Inoltre un MAIL PROGRAM e' usato per creare una lista di nomi e di indirizzi che per essere utilizzata deve essere immagazzinata anche questa. Infine per un completo utilizzo sara' necessario disporre di una stampante per le lettere, gli elenchi. Per questo motivo difficilmente potremo limitarci alla sola unita' centrale. Volendo sintetizzare un computer ha essenzialmente tre grandi capacita':

CALCOLO
SCELTA
COMUNICAZIONE

Fino a questo momento abbiamo parlato delle due prime capacita', vediamone ora la terza. La capacita' di comunicare con l' esterno e', in un computer, certamente la piu' complessa perche' e' necessario per prima cosa conoscere una serie di regole precise per l' invio e la ricezione dei dati, e poi adeguarsi a queste regole, cioe' metterle in pratica, renderle operative. Questa capacita' di comunicare con il mondo esterno (esterno al microprocessore naturalmente) sono veramente notevoli, tuttavia noi ci limiteremo per il momento a prendere in considerazione solo la possibilita' di leggere, scrivere e modificare dati e programmi su nastro che e' la periferica piu' usata per il suo costo, e solo dopo vedremo il disco.
La capacita' del processore di comunicare con la memoria RAM e ROM e con il video e la tastiera sono state abbondantemente trattate in precedenza.
Per quanto riguarda le altre periferiche collegabili ad un computer come: MODEM, RS-232, LINEE SERIALI E IEEE oltre a quanto altro si puo' collegare come Paddles, Joystick e Light-Pen, non possiamo dare che notizie

superficiali perche' esulano dallo scopo di questa guida e rimandiamo ad altri volumi di prossima (speriamo) pubblicazione e che riguarderanno come detto anche l' Hardware.

IL CONCETTO DI FILE

Sia su dischi che su cassette le informazioni sono immagazzinate come " FILES".
Per comprendere il concetto di FILE immaginiamo un comune schedario da tavolo entro il quale conserveremo, per esempio delle schede contenenti indirizzi di nostri amici, clienti o altro. Avremo quindi che lo schedario sara' il contenitore del nostro FILE, cioe' la cassetta o il dischetto. Non l' unita' a cassette ma il nastro stesso.
L' insieme delle schede sara' il FILE vero e proprio e che quindi potra' ampliarsi in rapporto alla grandezza fisica del contenitore.
E' abbastanza facile capire che le singole schede saranno i RECORDS, mentre i dati della scheda saranno i campi o FIELDS. Per un utente di computer il concetto di FILE deve essere considerato come di primaria importanza, ma non e' difficile da comprendere.
Quando si apre (OPEN) un file, tutte le informazioni ivi immagazzinate diventano accessibili e lo rimangono fino a quando non si chiude (CLOSE). Rifacendosi all' esempio iniziale e' come aprire il contenitore.
Quando un computer scrive un programma o dei dati su cassetta o disco, crea un nuovo file o aggiunge qualcosa ad uno vecchio. Un file puo' avere una lunghezza qualsiasi, limitata sola dalla capacita' della cassetta o del disco.
Si puo' creare un nuovo file senza scrivere niente dentro, cio' equivale ad avere una serie

di schede bianche senza alcun contenuto. Si possono avere numerosi files per ogni dischetto (dipende da che tipo di unita' CBM si sta adoperando), mentre non esiste limite per la cassetta. L' ammontare della memoria non ha nessun effetto sulla grandezza di un file di dati. Un file di dati puo' essere piu' grande della memoria disponibile del vostro computer. Aprendo un file di dati infatti si puo' leggere un solo carattere o piu' informazioni e passarle alla memoria centrale del computer.
Quando si scrive un file di dati, le informazioni che passano dalla memoria del computer a quella di massa possono essere, e di norma e' cosi',aggiunte a dati immagazzinati in precedenza sulla cassetta o sul floppy.

FILES PROGRAMMI

Ci sono due differenti tipi di files:

FILES PROGRAMMI
FILES DATI

Un file programmi, come e' implicito nel nome contiene un serie di comandi in Basic, in Assembler, in Pascal o qualsiasi linguaggio si stia usando, messi insieme come programma. Si crea un file programmi utilizzando un comando SAVE. Cioe' quando si usa il comando SAVE automaticamente viene creato sulla periferica un file PROGRAMMA.
Per creare un file di dati e' invece indispensabile usare piu' di un comando il che ci riporta al concetto di programma piu' o meno piccolo.
Un file puo' avere un nome, per cui il nome che assegnerete ad ogni file programma sara' posto in testa al programma.
Il discorso del nome e' riferito SOLO alla

cassetta perche', come vedremo si puo' anche registrare un gruppo di informazioni SENZA dare nessun nome. Non cosi' su disco.
Stante alle specifiche fornite dalle case costruttrici i computers CBM dovrebbero riconoscere nomi di files fino ad un massimo di 128 caratteri, ma solo i primi 16 caratteri sono visualizzati sullo schermo. Tuttavia non si riesce a capire bene come facciano per i 128 caratteri in quanto da nessuna parte del sistema operativo di cassetta o sulla directory del disco e' possibile trovare tanto spazio.
I nomi dei files disco possono avere 16 o meno caratteri per questo sara' un buon principio di restringere tutti i nomi di files a 16 caratteri o meno. L' ammontare di memoria del vostro computer ha effetto sulla grandezza massima di un PROGRAM FILE. Cio' e' perche' si crea un singolo file programma quando si salva (SAVE) un programma su cassetta o su disco.
Quando si carica un programma in memoria si carica l' intero contenuto del file programma. Non si puo' caricare una parte di un file programma in memoria. Per questo la grandezza massima di un file programma deve essere minore della capacita' di memoria programma del vostro computer.
La domanda e' come si faccia allora a caricare un programma molto grande. Se e' necessario caricare un programma molto grande e quindi non e' disponibile una memoria del computer sufficientemente vasta si puo' suddividere il programma in tanti sottoprogrammi, ognuno dei quali potra' essere contenuto nella memoria del computer. Quando ogni sezione del programma ha completato il lavoro, cioe' la sua esecuzione, semplicemente caricheremo la sezione successiva in memoria e la faremo girare. In questo modo si puo' eseguire l' intero programma. Successivamente descriveremo i passi necessari

per eseguire un programma in questo modo.
Un vantaggio dei files programma e' che il salvataggio ed il caricamento avviene tramite il DOS, cioe' il DISK OPERATING SYSTEM,' o per la cassetta, tramite il sistema operativo del computer. Sara' quindi necessario applicare un identificatore al file programma, tramite il suo nome o la sua locazione, per caricarlo nella memoria, ma e' l' unica operazione necessaria.

COMANDI PER I FILES PROGRAMMI

I comandi per i files programmi sono in verita' molto semplici ed essenzialmente si riducono a 3:

LOAD

SAVE

VERIFY

Il formato di LOAD e':

LOAD"nome del programma",d

d = il numero della periferica interessata. Nel caso della cassetta non e' necessario mettere il parametro d.

ESEMPIO

Caricare da cassetta il programma "MAILING LIST".

```
LOAD"MAILING LIST"
```

Il sistema rispondera' con un messaggio:

PRESS PLAY ON TAPE

e dopo aver eseguito l' operazione sul tasto indicato verra' visualizzato un:

OK

quindi il sistema iniziera' il caricamento del programma che sara' copiato dal nastro sulla memoria RAM del computer.
Il Sistema Operativo del computer genera automaticamente un' istruzione di OPEN (vedi dopo) usando l' indirizzo secondario appropriato per effettuare l' operazione di LOAD. Sulla periferica cosi' attivato viene iniziata una ricerca per trovare il programma il cui nome e' specificato nell' istruzione LOAD.
Dopo che il programma e' stato trovato esso viene letto automaticamente dalla periferica e caricato nella memoria partendo dall' indirizzo specificato nell' intestazione del file. Gli errori di lettura che possono verificarsi durante l' esame del primo blocco vengono automaticamente corretti dal secondo blocco perche' ricordiamo che le registrazioni sono fatte in doppio. Alla fine del ciclo viene eseguita una somma di prova o:**CHECKSUM**.
Se si ha un errore di Checksum o se siamo in presenza di un errore che non e' correggibile il sistema operativo visualizza un messaggio di:

? LOAD ERROR

ed arresta il caricamento del programma.

VERIFY

Questo comando ha la stessa sintassi del comando

LOAD:

VERIFY"nome del programma",d

In effetti l' istruzione di VERIFY e' un caso speciale di LOAD che dovrebbe venir eseguita dopo aver registrato qualsiasi programma. Il comando VERIFY fa si che il Basic esegua le stesse operazioni dell' istruzione LOAD, con la differenza che i dati non vengono caricati in memoria ma vengono confrontati con il contenuto della memoria. Se vengono incontrati degli errori, sia nel primo che nel secondo passo, il computer visualizzera' un messaggio:

? VERIFY ERROR

e sara' necessario registrare una seconda volta il programma in quanto la copia precedente non e' utilizzabile.

SAVE

Anche l' istruzione SAVE provoca un' operazione di apertura e chiusura automatica di un file. Ha il seguente formato:

SAVE"nome del programma",d

dove d e' il numero della periferica. Se la periferica e' una unita' a nastro il Sistema Operativo del Computer inizia automaticamente a registrare, naturalmente dopo aver premuto i relativi tasti, un' intestazione ed apre un file su nastro con un nome appropriato. Se il dispositivo e' un' unita' a disco, viene inviato uno speciale messaggio di apertura che sta ad indicare che il computer sta inviando un file programma. Immediatamente dopo il programma

viene scritto direttamente dalle sue locazioni di memoria o sul nastro o sul disco.

DATA FILES

Un DATA FILE o un files di dati come dovrebbe essere implicito nel nome, contiene informazioni che devono essere interpretati come dati in opposizione a comandi di programma. I files dati sono creati, scritti e letti tramite i programmi, cioe' non possono essere scritti o letti direttamente come un file programma per mezzo delle istruzioni LOAD e SAVE.

RECORDS E FIELDS

I data files sono divisi in records che a sua volta sono suddivisi in FIELDS o campi. Un singolo field contiene informazioni che possono essere rappresentate tramite il nome di una singola variabile.
Per questo motivo un field puo' contenere un numero intero, un numero in virgola mobile o una singola variabile stringa. Un record contiene uno o piu' field. I records di solito rappresentano unita' d' informazioni ripetitive entro il file, ma puo' non essere sempre cosi'. Prendiamo per esempio una mailing list. L' intera mailig list puo' essere considerata come un singolo file di dati. Ogni nome e indirizzo entro la mailing list sara' un record entro il file. Un esempio di file che gestisce un indirizzario potrebbe essere caricato con i seguenti dati. In questo caso ogni record conterra' 5 campi:

Il cognome
Il nome
La via
Il CAP
La citta'

Un file puo' contenere uno o piu' record. Ogni record puo' contenere uno o piu' fields. Il numero di records in un file e la massima lunghezza di un record varia con il tipo di file come descriviamo successivamente. Tuttavia in pratica la grandezza di un file e' limitata solo dalla capacita' di memoria di massa. Nessuna restrizione invece alla lunghezza di un record su cassetta. Un record puo' avere una lunghezza qualsiasi che entri pero' nella lunghezza del nastro e questo per il semplice motivo che non puo' essere diviso in due nastri fisici.

FILES LOGICI E UNITA' FISICHE

Si usa il termine "INPUT/OUTPUT PROGRAMMING" per descrivere la logica di programmazione che consente il trasferimento dati fra il computer e le unita' esterne.
I dischi, le cassette e le stampanti sono quindi unita' fisiche esterne.
Per consentire una qualsiasi operazione di INPUT/OUTPUT (ingresso/uscita) il programma deve identificare l' unita' fisica esterna alla quale si deve accedere.
Pensiamo il problema in termini di programmazione. Questo concetto e' facile da capire se si pensa alla tastiera ed al video come unita' esterne rispetto al computer propriamente detto, come in effetti esse sono. Quando viene eseguito un comando INPUT i dati che abbiamo immesso con la tastiera sono

specificati in un parametro di input. Quando il comando :

```
10 INPUT A
```

e' eseguito, alcuni numeri che l' operatore fa entrare attraverso la tastiera sono assegnati a una variabile (in virgola mobile). Nello stesso modo il comando di PRINT visualizzera' variabili o costanti. Cosi' il comando PRINT:

```
20 PRINT A
```

prende i valori assegnati alla variabile in virgola mobile A e mostra questo valore sullo schermo. Quando viene eseguito un comando di INPUT l' unita' fisica esterna e' vista come nel caso precedente e' stato per la tastiera. Quando viene eseguito invece un comando di PRINT l' unita' fisica esterna viene vista come il video. La programmazione degli INPUT/OUTPUT diventa molto piu' complessa quando i dati sono trasferiti da/a e dalla cassetta, il disco, la stampante e altre unita' fisiche esterne al di fuori della tastiera e del video.
Per queste piu' complesse operazioni di INPUT o OUTPUT dovrete prima di tutto aprire un :

CANALE DI COMUNICAZIONE

tra il programma e l' unita' fisica selezionata. Dopo aver eseguito l' operazione richiesta di INPUT/OUTPUT dovrete richiudere il canale. Il Basic del CBM identifica canali singoli usando un numero di canali che puo' andare da 0 a 255. Si apre un canale usando il comando :

OPEN

I Parametri di questo comando identificano l'

unita' fisica alla quale si deve accedere, mentre per la natura di questo accesso vedremo in dettaglio nelle parti seguenti. Fino a che il canale non sia chiuso, ogni comando di INPUT/OUTPUT necessita solo che sia specificato il numero del canale per descrivere completamente la natura della operazione di input o di output. Ogni unita' fisica ha di per se un solo numero di riconoscimento fisico.Questo numero e' usato come un parametro quando si apre un canale per identificare l' unita' fisica alla quale si vuole accedere. I numeri di canale sono per questo riportati frequentemente come "LOGICAL FILES NUMBERS" o "LOGICAL UNITS NUMBERS".
Il nome LOGICAL FILE descrive un canale molto accuratamente, perche' un canale stabilisce un legame tra un programma e un file di dati. I FILES LOGICI sono un concetto di programmazione. Si puo' iniziare una qualsiasi operazione di I/O usando un comando di OPEN. Uno dei parametri del comando OPEN e' il canale o il numero di FILE logico. Gli altri parametri identificano l' unita' fisica, i dati ai quali si deve avere accesso ed il mezzo in cui occorre questo accesso.
Dopo che una operazione di input o output e' stata completata bisogna eseguire un comando CLOSE che richiudera' il canale. Il comando CLOSE richiede solo un parametro: **IL CANALE O NUMERO DI FILE LOGICO**
Questo numero di file logico unisce quindi un comando CLOSE ad un comando OPEN.
Tra un comando OPEN ed un CLOSE tutti i comandi di I/O usano un canale o un logical file number per identificare l' unita' alla quale si deve accedere e l' operazione che deve essere eseguita.
Il LOGICAL FILE NUMBER mette in relazione i comandi di :

OPEN
CLOSE
GET#
PRINT#
INPUT#

Con qualsiasi altro.
Fino a quando state usando un numero di file logico in un comando, non potete riutilizzare lo stesso numero di file logico per fissare un diverso canale di I/O fino a che il LOGICAL FILE non sia chiuso. Se lo farete il Basic del computer rispondera' con un :

FILE OPEN ERROR

D'altra parte nessun altra limitazione e' presente nel metodo di assegnare un numero di File Logico entro il vostro programma. Il numero di DEVICE o di periferica identifica l' unita' fisica alla quale il computer inviera' i suoi dati o dalla quale li ricevera'. Il numero di device appare come un parametro nel comando OPEN. Ogni unita' fisica che possa comunicare con un computer CBM ha assegnato in permanenza un numero di DEVICE. Non appena venga trovato un numero di device in un comando OPEN il computer attiva un' appropiata logica elettronica per stabilire una comunicazione con l' unita' specifica identificata nel numero di device. Teoricamente sono disponibili 256 numeri di periferiche in un range compreso fra 0 e 255. Tuttavia solo i numeri di device fra 0 e 30 sono correntemente usati. In aggiunta alla definizione del numero di device, molte unita' fisiche rispondono ad un vasto gruppo di indirizzi secondari.

INDIRIZZO SECONDARIO

Oltre ad avere un numero di unita' fisica a molte periferiche puo' essere assegnato un indirizzo secondario o SECONDARY ADRESS. L' indirizzo secondario e' un comando che parte dal computer e che dice all' unita' fisica quale operazione deve prepararsi ad eseguire.
Non dovreste impegnarvi in uno studio particolare degli indirizzi secondari, perche' successivamente quando descriveremo i programmi di I/O in dettaglio la funzione di indirizzo secondario diventera' familiare ed ovvia per il suo frequente uso. Il programma seguente illustra molto bene l' uso dei parametri nei comandi di I/O.

```
100 OPEN 4,1,2,"MAILING LIST"
200 PRINT#4,CN$
210 PRINT#4,NO$
220 PRINT#4,VP$
230 PRINT#4,CA$
240 PRINT#4,LO$
300 CLOSE4
```

I 5 comandi di PRINT# che appaiono nelle linee dalla 200 alla 240 scrivono 5 parti di nome ed indirizzo in un file chiamato :**MAILING LIST** nastro dell' unita' a cassetta.
Tutte le volte che si incontra un comando di PRINT# il computer sa cosa deve fare perche' controlla il numero di File logico che appare dopo il carattere #. Nel programma questo numero di File logico e' 4, percio' nel comando di OPEN e' specificato il file logico 4 che descrive la natura dell' operazione. Questo comando di OPEN e' presente nella linea 100 del nostro programma. Se il computer non dovesse trovare un comando di OPEN con il richiesto

numero di unita' logica, questi non potrebbe mettere in funzione le operazioni di I/O poiche' non saprebbe cosa fare.
Nel programma c'e' un comando di OPEN sul File Logico n. 4. Questo comando specifica l' unita' logica n. 1 che appunto sta ad indicare che e' selezionata la cassetta. L' indirizzo secondario e' 2 perche' in questa occasione e' possibile scrivere sulla cassetta del drive 1 ma non e' possibile leggerci. Quando questa operazione e' chiusa verra' scritto un fine nastro sulla cassetta per prevenire che un qualsiasi dato possa essere successivamente aggiunto. Il comando OPEN specifica inoltre che il Data file al quale si deve accedere ha il nome MAILIG LIST.
Sulla linea 300 e' presente un comando di CLOSE (chiudi). Questo comando specifica il numero 4 come File Logico, di conseguenza tutto quanto e' stato aperto con il comando OPEN nella linea 100 sara' chiuso con questo comando alla linea 300. Poiche' il comando OPEN alla linea 100 specifica un numero di indirizzo secondario 2, il comando CLOSE alla linea 300, quando sara' eseguito causera' una EOF (END OF FILE) sulla cassetta.
In questo modo il file logico n 4 che e' presente nei comandi dalle linee 200 alla linea 300 congiunge questi comandi con un comando OPEN alla linea 100. Parametri addizionali appaiono sui comandi OPEN alla linea 100 per descrivere le operazioni che devono essere eseguite.
Prima di procedere oltre con la programmazione vediamo alcuni concetti, in particolare per le variabili di controllo che sono essenziali per la gestione delle periferiche.

FISICAL UNIT STATUS

Una stampante puo' ricevere informazioni da un computer, cioe' si possono preparare delle stringhe da far stampare su una stampante, tuttavia da una stampante i dati non possono passare ad un computer. Per questo motivo non e' necessario specificare il numero di indirizzo secondario quando si esegue un comando di OPEN su una periferica tipo stampante.
Al contrario una cassetta puo' ricevere dati dal computer o trasmetterglieli, per cui l' indirizzo secondario usato nel comando OPEN che inizializza la cassetta dovra' specificare se l' operazione e' di lettura o di scrittura.
Quando si esegue un comando di PRINT#, GET# o INPUT# e' necessario fare attenzione a quello che si vuol fare. In altre parole non sara' possibile eseguire dei comandi di INPUT o di GET quando l' unita' a cassetta sara' stata preparata solo per operazioni di scrittura. Se questo dovesse avvenire avremo una registrazione di errore di STATUS. L' unita' fisica riporta l' informazione sullo status di seguito ad ogni operazione di INPUT o di OUTPUT quando questa sia stata eseguita con successo o con insuccesso. In pratica tutte le volte che si accede ad una unita' periferica, considerando pero' in questo caso come periferiche anche la tastiera ed il video, viene attivato un registro di 8 bit che e' appunto: **REGISTRO DI STATUS.** Questo registro ha come riferimento la variabile Basic ST. Per esempio il comando **10 X= ST** assegnera' al registro di status il valore della variabile X.

MANIPOLAZIONE DI DATI SU CASSETTA

Passiamo ora a descrivere i passi di programma necessari per la manipolazione dei files su cassetta. Potete programmare il computer per scrivere dati su cassetta o per rileggerli, ma non potete programmare il movimento fisico della cassetta. E' importante che comprendiate il modo in cui opera fisicamente il DRIVE, cioe' l' unita'. In altre parole dovete tenere presente che per eseguire operazioni sulla cassetta non sarete mai in grado di manipolare il movimento fisico del nastro. In effetti questo discorso non e' completamente vero perche' si puo' programmare, ad esempio agendo su determinati registri, l' arresto del motore. I files sono immagazzinati in modo sequenziale su nastro, cioe' uno di seguito all' altro. Un HEADER cioe' una testata, precede il primo file e un fine nastro (EOT) segue l' ultimo file. Ogni fine di file e' segnato da un EOF. La testata e' scritta automaticamente all' inizio del nastro. A questo punto potete notare che l' attivita' della cassetta o almeno questa parte di attivita' della cassetta, non vi riguarda. In altre parole l' esistenza di un HEADER viene scritta automaticamente, cioe' non ha bisogno di vostre operazioni. Il computer puo' trovare i File mentre il nastro sta girando piano, cioe' alla velocita':**PLAY** ma non e' in grado di trovarli quando sta girando in FF, cioe' in **FAST FORWARD** e questo perche' durante la fase di LETTURA/SCRITTURA non e' in contatto fisicamente con il nastro.
Il computer non puo' eseguire una operazione di riavvolgimento diretto veloce ne puo' trovare niente sulla cassetta mentre il nastro si sta riavvolgendo. Si deve iniziare il movimento sulla cassetta manualmente premendo il tasto

relativo in seguito ad una istruzione fornita dalla unita' centrale. Si raccomanda di non premere nessun tasto prima che un messaggio venga visualizzato. In seguito potremo comportarci diversamente dopo aver pero' preso un po' di pratica nelle operazioni. Esaminiamo ora l' impatto sulle operazioni dell' unita' a cassetta.
Quando si stanno scrivendo dati sulla cassetta, il nastro deve essere correttamente posizionato ad inizio scrittura e cio' e' messo sotto la responsabilita' dell' operatore. A questo punto e' necessario ricordarsi che se non si posiziona correttamente il nastro e' facile avere delle sovrascritture. Inoltre se la parte iniziale trasparente e' posizionata sulla testina di scrittura, l' unita' cerchera' di scrivere le informazioni che pero' non saranno registrate. E' importante ricordarsi di questo perche' il computer non e' capace di distinguere la superfice magnetica dalla superfice non magnetica. Il metodo che consente la massima sicurezza e' di iniziare a scrivere su nastro vergine o su una cassetta i cui dati non servano piu' e di posizionare il nastro all' inizio della superfice magnetica. Si possono cosi' tranquillamente scrivere records e files uno dietro l' altro fino al termine fisico del nastro stesso. Il Sistema Operativo dell' unita' centrale nella parte relativa all' uso dell' unita'a a dischi si assicurera' che venga lasciato uno spazio sufficiente fra la fine di un record o di un file e l' inizio del successivo, per cui non sara' necessario che l' operatore si occupi di questo. Quando si leggono files di dati gia' registrati, e' necessario assicurarsi che il nastro sia riavvolto fino all' inizio del primo file che si vuole rileggere. Il computer puo' trovare un qualsiasi definito file di dati purche' questo sia DOPO il

punto in cui l' abbiamo fatto partire, ma non puo' certo tornare indietro a cercarselo da solo. Non si deve mai cercare di riscrivere anche una piccola parte di file su nastro perche' l' operazione e' troppo rischiosa.
Supponiamo per esempio di aver immagazzinato su un file cassetta 10 nomi ed indirizzi e che si desideri variare il quinto nome ed il relativo indirizzo. Teoricamente si potrebbe leggere i primi 4 nomi ed indirizzi e questa operazione ci dovrebbe lasciare il nastro posizionato all' inizio del quinto nome. Si potrebbe quindi scrivere il nuovo quinto nome sul vecchio. In pratica e' meglio non farlo.
Infatti l' unita'a a cassetta non e' molto precisa e c' e' una buona probabilita' che il nuovo nome ed indirizzo sia scritto un po' prima o un po' dopo del vecchio. Infatti, a parte il fatto che dovrebbero essere della stessa identica lunghezza e questo risulta difficile da ottenere, e' sufficiente che un solo carattere vada fuori posto, cioe' o troppo prima o dopo, che non saremo in grado di rileggere i dati. Per aggiornare quindi un file di dati e' necessario caricare il file stesso nella memoria del computer, aggiornarlo e quindi riscriverlo.
Potrebbe sembrare che la trattazione di questi problemi sia stata troppo lunga, ma per esperienza diretta e per le numerose lettere che ci sono pervenute, possiamo assicurarvi che i guai sulla registrazioni su nastro possono far perdere piu' tempo che non la scrittura stessa dei programmi.

IL FORMATO DEI FILE SU CASSETTA

Ogni field numerico deve essere seguito da un ritorno carrello effettuato con l' inserzione del carattere (CHR$(13)). Percio' un file che

consiste solo di campi numerici dovrebbe essere visto come una sequenza di numeri separati da un carattere di ritorno carrello come e' illustrato nell' esempio seguente dove N sta' per il numero e CR per il carattere di ritorno carrello:

N--CR--N--CR--N--CR

Per questo all' interno di un file di dati puramente numerico non esiste nessuna ripartizione di fields in record o distinzione fra i vari records. E' interamente demandato quindi alla logica del programma la formazione di record intesi come sequenza di fields sempre che questo sia necessario. Le variabili stringa da memorizzarre possono invece essere divisi facilmente in fields e questi eventualmente raggruppati in records.
Si possono usare i due punti (CHR$(44)) per separare i fields entro un record, mentre un ritorno carrello (CHR$(13)) seguira' l' ultimo field del record stesso.Nell' esempio seguente viene mostrato la struttura fisica di registrazione su nastro di un file che contiene solo variabili stringa con 5 file per record. Ricordiamo che, come nell' esempio precedente CR sta per il ritorno carrello mentre S per la stringa:

CR--S--:--CR--S--:--CR--S--:

Se si usano i due punti ed il ritorno carrello come separatori per dividera il file stringa in Fields e record come illustrato precedentemente, allora tutti i fields di ogni record devono essere letti con un singolo comando di INPUT#.
Prendendo come esempio il programma di gestione indirizzi si vede come la logica di programmazione richieda essa stessa che vari fields siano organizzati in record in maniera

evidente. Non e' necessario insegnare come un programmatore debba vedere che ogni nome ed indirizzo diventa un record, mentre parte del nome e dell' indirizzo devono essere trattati come singolo field. Le strade per dividere un nome ed indirizzo nei singoli fields sono numerose e in pratica, purche' risponda allo scopo che ci siamo prefissi una strada vale l' altra. Ora partendo da un semplice programma cercheremo di approfondire la sintassi dei comandi e tramite piccole continue modifiche spiegheremo cio' che e' concesso e cio' che non e' concesso fare. Digitiamo il seguente programma:

```
10  OPEN1,1,1
20  FORI=1TO10
30  PRINT#1,I+100
40  NEXT
50  CLOSE 1
60  STOP
70  OPEN1
80  FORI=1TO10
90  INPUT#1,J
100 PRINTJ
110 NEXT
120 CLOSE1
132 STOP
```

Il comando OPEN alla linea 10 apre il file logico 1, selezionando l' unita' a cassetta per una operazione di scrittura.
Il ciclo di FOR-NEXT alle linee 20,30 e 40 scrivono 10 numeri su nastro. I numeri sono seguiti da un carattere di ritorno carrello per far si che il comando di PRINT# alla linea 30 forzi un ritorno carrello ad ogni scrittura su nastro, allo stesso modo che un comando di PRINT forza un a capo sullo schermo. Il file logico e' chiuso alla linea 50. La struttura fisica della

registrazione su nastro sara' la seguente:

CR--1--CR--2--CR--3...10--CR

I comandi dalla linea 70 alla linea 120 leggono e visualizzano i dieci numeri che erano stati scritti su nastro dai comandi che vanno dalle linea 20 alla linea 50. Digitate il programma e salvatelo su una cassetta programmi. Prendete poi un cassetta vergine ricordando le precauzioni precedenti. Assicuratevi che nessun tasto sia premuto e digitate RUN. Verra' visualizzato il seguente messaggio:

PRESS PLAY AND RECORD ON TAPE #1

Premere i tasti indicati sulla unita' a cassette. Il computer visualizzara' un : OK sotto la frase precedente. Il nastro comincera' a girare mentre i numeri da 101 a 110 saranno registrati. Dopo che i 10 numeri sono stati scritti la cassetta si ferma e sullo schermo viene visualizzato il seguente messaggio:

BREAK IN 60

con il cursore che lampeggia sotto la scritta Ready. Il comando di STOP alla linea 60 ha causato l' interruzione. Premere ora il tasto di STOP della cassetta per far rialzare i tasti di PLAY e di RECORD. Riavvolgere la cassetta con il tasto REWIND e ripremere il tasto STOP per far rialzare il REWIND. Infatti anche se quest' ultimo si rialza da se la tensione sul nastro puo' far si che questo si spezzi. Eseguiamo la seconda parte del programma digitando GOTO 70. Sara' visualizzato il messaggio:

PRESS PLAY ON TAPE 1

Se fate attenzione a premere SOLO il PLAY sulla casseta ed il computer rispondera' con un OK dopo l' operazione il nastro comincera' a girare e dopo aver superato la parte bianca della cassetta e letto i 10 numeri scritti prima visualizzera' su una colonna verticale dello schermo:

```
101
102
103
104
105
106
107
108
109
110
BREAK IN 130
```

Il messaggio finale e' dovuto all' esecuzione del comando STOP che e' presente alla linea 130 del nostro programma. Se avete dimenticato di riavvolgere il nastro prima di digitare GOTO 70,l' unita' cerchera' per tutto il nastro, che si e' detto doveva essere vergine per questo esempio, e non trovando niente andra' fino alla fine fisica del nastro. Se siete incorsi in questo errore dovete per prima cosa premere il tasto STOP della unita' a cassetta e poi fermare il programma premendo il RUN/STOP del calcolatore. Successivamente riavvolgere il nastro, ma prima di far ripartire il programma dovete eseguire una operazione di chiusura in forma diretta digitando CLOSE 1 e quindi ripartire con un GOTO 70. Chiarito il primo errore che si puo' commettere e che e' piu' frequente di quanto non si creda passiamo ad effettuare delle piccole modifiche al nostro programma. Listiamo il programma ed aggiungiamo

al comando PRINT nella linea 100 un punto e virgola in modo da avere: 100 PRINT J.Riavvolgiamo il nastro e digitiamo di nuovo GOTO 70. Dopo avere eseguito l' operazione di lettura come precedentemente descritto sullo schermo apparira':

101 102 103 104 105 106 107 108 109 110
BREAK IN 130

A titolo sperimentale proviamo a cambiare i comandi dalla linea 80 alla 110 in modo tale che i dieci numeri siano inseriti usando un solo comando di INPUT. Questo esempio serve a dimostrare che non esiste nessuna differenza nella lettura dei dieci numeri sia eseguendo il comando INPUT# con 10 variabili come suoi parametri che eseguendo lo stesso comando con una sola variabile ma 10 volte. Continuiamo il nostro ciclo di esperimenti modificando il sistema di separazione all' interno del file numerico. Proviamo ora a cambiare la prima parte, cioe' quella relativa alla scrittura.Il programma sara' come segue:

```
10  OPEN1,1,1
20  FOR I =1 TO 10
30  M(I) =I + 100
40  NEXT
45  C$=CHR$(59)
46  PRINT#1, M(1);C$; M(2);C$; M(3);C$; M(4);C$;
M(5)
47  PRINT#1, M(6);C$; M(7);C$; M(8);C$; M(9);C$;
M(10)
50  CLOSE 1
60  STOP
70  OPEN 1
80  FOR I =1 TO 10
90  INPUT#1, J
100 PRINT J
```

```
110 NEXT
120 CLOSE 1
130 STOP
```

dove CHR$(59) rappresenta un punto e virgola(;). Eseguiamo nuovamente l' operazione di scrittura nastro ricordando di far apparire sulla finestrella la prima parte della superfice magnetica. Dopo che sara' apparso il solito: PRESS PLAY AND RECORD ON TAPE #1 eseguiamo le operazioni abituali e dopo la registrazione avremo:**REAK IN 60**.
Riavvolgiamo il nastro e digitiamo il GOTO 60 per far eseguire l' operazione di lettura. Dopo aver premuto il tasto PLAY sulla cassetta sara' visualizzata un scritta:

FILE DATA ERROR IN 90

Cio' vorra dire che i dati non sono stati letti correttamente. Perche'?.
Perche' non si puo' usare nessuna altra forma di punteggiatura e quindi di separazione nel trattamento di dati numerici all' infuori del ritorno carrello. Si possono invece usare i due punti o il ritorno carrello per separare i campi in una stringa. Proviamo a cambiare il programma come segue:

```
 5  DATA UNO, DUE, TRE,  QUATTRO,  CINQUE,  SEI,
SETTE, OTTO, NOVE, DIECI
10  OPEN1,1,1
20  FOR I =1 TO 10
30  READ M$ (I)
40  NEXT
45  C$=CHR$(44)
46   PRINT#1,  M$(1);C$;   M$(2);C$;   M$(3);C$;
M$(4);C$; M$(5)
47   PRINT#1,  M$(6);C$;   M$(7);C$;   M$(8);C$;
M$(9);C$; M$(10)
```

```
50  CLOSE 1
60  STOP
70  OPEN 1
80  FOR I =1 TO 10
90  INPUT#1, J$
100 PRINT J$
110 NEXT
120 CLOSE 1
130 STOP
```

Riavvolgiamo quindi il nastro e eseguiamo la prima parte di scrittura del programma. Vedremo che i dati saranno scritti correttamente e sara' visualizzata la scritta: BREAK IN 60 seguita da READY.
A questo punto riavvolgiamo la cassetta ed eseguiamo il solito GOTO 70. Dopo aver premuto il tasto PLAY avremo la seguente visualizzazione
UNO SEI
mente sotto sullo schermo:

STRING TOO LONG ERROR IN 90

Cosa e' successo?
Il problema sta nel comando INPUT# della linea 90. Un comando INPUT# leggera' tutti campi della stringa fino al primo ritorno carrello. Quindi le variabili da M$(1) a M$(5) sono in INPUT alla prima esecuzione del comando INPUT# alla riga 90. Tuttavia solo il valori di M$(1) e' stato letto da J$ perche' la virgola e' interpretata come un separatore di campo e non come un segnale di termine. La seconda volta il comando INPUT# alla linea 90 e' eseguito, da M$(6) fino a M$(10) sono in input, poiche' questi sono due campi situati fra due ritorni carrello. Ancora una volta solo M$(6) e' assegnato a J$ poiche' la virgola e' interpretata come campo. La terza volta che il comando INPUT# alla linea 90 viene eseguito non ci sono piu' dati da leggere e

viene segnalato pertanto un errore. Per risolvere questo problema e' necessario eseguire i comandi di INPUT# con lo stesso numero di variabili presenti nel comando PRINT#. Consideriamo il seguente programma:

```
 5  DATA UNO, DUE, TRE,  QUATTRO,  CINQUE,  SEI,
SETTE, OTTO, NOVE, DIECI
10  OPEN1,1,1
20  FOR I =1 TO 10
30  READ M$(I)
40  NEXT
45  C$=CHR$(44)
46   PRINT#1,  M$(1);C$;   M$(2);C$;   M$(3);C$;
M$(4);C$; M$(5)
47   PRINT#1,  M$(6);C$;   M$(7);C$;   M$(8);C$;
M$(9);C$; M$(10)
50  CLOSE 1
60  STOP
70  OPEN 1
80  INPUT#1, N$(1), N$(2), N$(3), N$(4), N$(5)
90  INPUT#1, N$(6), N$(7), N$(8), N$(9), N$(10)
100 FOR I = 1 TO 10
105 PRINT N$(I);" ";
110 NEXT
120 CLOSE 1
130 STOP
```

Se non commetteremo nessun errore nel posizionamento del nastro avremo, stavolta correttamente: UNO DUE TRE QUATTRO ecc. sul video seguiti da: BREAK IN 130
E' probabile che siamo stati noiosi, ma vi assicuriamo che la manipolazione dei dati su cassetta o su disco e' veramente la parte piu' importante della programmazione. Per questo vi consigliamo di continuare con gli esperimenti e tutte le prove che riterrete necessarie fino a prendere assoluta confidenza con i metodi descritti.

LETTURA DI DATI DA CASSETTA

Ci sono tre passi di programma necessari per leggere un file dati da cassetta:

APRIRE IL FILE con un OPEN
LEGGERE IL FILE con INPUT#
CHIUDERE IL FILE con un CLOSE

Un file di dati deve essere aperto per la lettura utilizzando lo stesso nome (NAME FILE) che era stato adoperato per scriverlo.
Mentre puo' essere assegnato un diverso numero di file logico, l' indirizzo secondario deve essere posto a 0 appunto per l' opzione di lettura. Ricordiamo come regola generale:

```
SCRITTURA:    OPEN1,1,2,"DATA"
LETTURA:      OPEN1,1,0,"DATA"
```

Sono disponibili due comandi di lettura da cassetta: **INPUT# e GET#**. Per leggere un campo (FIELD) numerico o una stringa useremo il comando INPUT#.
Il comando GET# leggera' invece un carattere per volta. Ricordiamo poi di eseguire un CLOSE dopo che il file e' stato letto. E naturalmente chiudiamo lo stesso file logico che e' stato aperto. per cui nell' esempio di prima CLOSE 1.
Un buon sistema di chiudere un file e' quello di controllare l' esistenza di un carattere EOF (end-of-file) attraverso il registro di STATUS.
Quando un file viene scritto, un carattere di EOF viene posto alla fine del file. Quando viene letto un carattere di EOF, il registro di status assume il valore di 64 ed il file puo' essere chiuso. Si puo' controllare con questo semplice comando inserito alla fine della lettura:
IF ST=64 THEN CLOSE 1. Cioe' quando lo status e'

64 il file viene chiuso.
Precedentemente abbiamo scritto un programma per scrivere una serie di numeri da 1 a 10 in un file dati su cassetta chiamato. Ora scriveremo un programma per leggere i dieci numeri dal file di dati NUMBERS e visualizzarli sullo schermo.

```
10 PRINT"** LETTURA DI DATI NUMERICI"
15 PRINT
20 PRINT"INSERIRE LA CASSETTA E PREMERE IL
RETURN QUANDO SIETE PRONTI":
25 PRINT
30 GET A$:IFA$=""THEN 30
40 PRINT"APERTURA":OPEN1,1,0 "NUMERI"
45 PRINT
50 FOR I=1 TO10
60 INPUT #1,N
70 PRINT N
80 NEXT I
90 PRINT"CHIUSURA": CLOSE 1
100 END
```

Il comando INPUT# legge un campo per volta. Le prime 3 linee di del programma dicono all' utente come deve operare. I comandi sono identici a quelli del programma di scrittura. Alla linea 30 c'e' un loop di attesa che da all' operatore il tempo di montare il nastro sulla cassetta. Dopo il montaggio fisico del nastro, premere RETURN e il programma passa alla linea successiva. Prima che ogni dato sia letto, il file deve essere aperto. I comandi alla linea 40 apriranno il FILE#1,sulla periferica 1, con l' indirizzo secondario 0 per l' operazione di lettura sul file con il nome NUMBERS. Nelle linee da 50 a 80, il cuore del programma, con il ciclo FOR-NEXT vengono letti i primi 10 dati dal nastro e visualizzati sullo schermo. Il comando INPUT#1 della linea 60 legge un numero per l' esecuzione. Dopo che i dati sono stati letti il

file viene chiuso alla linea 90, mentre alla linea 100 si trova un END che ricordiamo pero' e' opzionale. Il comando INPUT# puo' anhe leggere campi che contengano variabili stringa. Come abbiamo visto in precedenza con il programma di numeri alfabetizzati, invece dei numeri nella loro rappresentazione decimale li abbiamo scritto in forma letterale. Per leggere le stringhe, in questo caso, e' sufficiente una piccola modifica al programma di lettura precedente. Come possiamo vedere dal listato che segue sono necessari solo dei cambiamenti alla linea 40 per fargli rileggere un file diverso da quello di prima ed alla linea 60 per far rileggere una stringa anziche' un numero. Cambieremo pertanto:

```
60 INPUT#1,N   con    60 INPUT#1,N$
70 PRINTN
```

APERTURA O OPEN

Si deve aprire un comando OPEN per aprire un file di dati qualsiasi operazione si desideri eseguire. Il formato di questo comando e':

```
OPEN N,D,S,Nome del File
```

Cioe' apri il file logico N, seleziona sulla periferica D il file di dati scelto con "Nome del file" e predisponi per eseguire l' operazione specificata con l' indirizzo secondario S. Si puo' usare il comando OPEN con una qualsiasi combinazione di questi parametri. N e' il solo parametro che deve essere presente, mentre se D e' assente viene ritenuto uguale a 1. Se e' assente S viene assunto per 0, mentre se manca il nome del file verra' selezionato il primo file che si incontra sulla cassetta. Quando viene eseguito un comando OPEN su

cassetta per la lettura di dati, verra' visualizzato il seguente messaggio: **PRESS PLAY ON TAPE** e dopo che il tasto della cassetta sara' premuto sara' visualizzato: **OK.**
Il computer incomincera' allora a leggere il nastro. Nell' ipotesi che il comando sia dato in modo immediato e che il file cercato non sia il primo sul quale si e' posizionata la testina di lettura, avremo la visualizzazione dei seguenti messaggi:

SEARCHING FOR (Nome del file)
FOUND Test
FOUND Ross
FOUND Chess
FOUND

dove i primi tre stanno ad indicare i nomi di files incontrati, mentre il quarto indica che si e' incontrato un file senza nome. In modo programma invece questo blocco di messaggi non sara' visualizzato. Quando il comando OPEN viene eseguito per una operazione di scrittura il computer visualizzera' il seguente messaggio: PRESS PLAY & RECORD ON TAPE . Anche questo messaggio sara' seguito da un OK quando vengano premuti i tasti. Il computer scrivera' il TAPE HEADER, poi si fermera' in attesa di dati. Vediamo alcuni esempi di utilizzo del comando OPEN commentandoli:

OPEN 1 Viene aperto il file logico 1. Nessuna periferica e' specificata per cui come periferica sara' assunta la cassetta n. 1. Non essendo inoltre specificato nessun indirizzo seconadrio il computer si prepara ad una operazione di indirizzo seconadrio 0 cioe' di lettura. E dato che nemmeno il nome del file e' specificato, verra' letto il primo file che incontra su cassetta.

OPEN 1,1 Come per il precedente, ma in questo caso e' specificata la periferica.
OPEN 1,1,0,"data" Come sopra ma con una apertura per leggere il file di nome "data" sulla cassetta.

OPEN 3,1,2 Apre il file logico 3 per la cassetta 1.Scrive un nuovo file e un carattere End of Tape alla fine del file. Il file che si registra non ha nessun nome.

OPEN 3,1,2,"Paolo" Come sopra ma questa volta con un file di nome PAOLO.

CHIUSURA DI UN FILE

I comandi di apertura e di chiusura di un file sono fra se strettamente correlati. Ricordiamoci che il comando CLOSE deve essere l' ultimo comando che si da nella sequenza logica della programmazione e che mentre se non si esegue l' apertura di un file e' immediatamente tangibile che non si puo' accedere ad esso, il fatto di usare il comando CLOSE e' lasciato alla accortezza del programmatore, perche' il sistema non dara' alcuna segnalazione. Il formato del comando e':

CLOSE N

dove N e' l' unico parametro da specificare e deve corrispondere al numero di file logico precedentemente dichiarato nel comando OPEN. Ricordiamo che quando e' stata eseguita un' operazionedi chiusura su un file dopo la lettura non sono piu' consentiti accessi per la lettura sullo stesso file, per cui sar' necessario riaprirlo. Non e' indispensabile eseguire la

chiusura di un file dopo la lettura, tuttavia non sara' cattiva pratica di programmazione prendere costante confidenza con questa operazione che in altri casi e' invece indispensabile. E' invece ASSOLUTAMENTE necessario eseguire la chiusura di un file dopo una operazione di scrittura. Infatti i dati non vengono mai trasferiti direttamente dalla memoria centrale del computer al nastro, ma passano attraverso un BUFFER, cioe' attraverso una zona polmone che provvede a comunicare i dati stessi a gruppi di 192 Bytes. Quando questo BUFFER e' stato riempito allora i dati passano al nastro. Ma solo allora, oppure quando si esegue un CLOSE per chiudere appunto il file al quale stiamo riferendoci. Per questo motivo accade quasi sempre che se al termine di un' operazione di scrittura non si esegue la chiusura un po' di bytes che quasi sicuramente sono sul nastro vengono persi. Inoltre quando si esegue una chiusura di un file nastro dopo un' operazione di scrittura, verra' anche inviato un carattere di fine file o END OF FILE (EOF) che verra' quindi scritto sul nastro stesso. Al sistema infatti necessita questo carattere di separazione fra un file e il successivo. Senza di questo infatti il computer potrebbe, successivamente in fase di lettura, continuare a leggere i dati del file successivo sul quale magari siamo andati a scrivere sopra. E' importante notare infatti che mentre in fase di input dati siamo certi della quantita' di dati da scrivere, altrettanto non avviene in fase di rilettura.

Quando si chiude un file preventivamente con un INDIRIZZO SECONDARIO 2, su cassetta verra' scritto un fine nastro END OF TAPE (EOT) alla fine del file stesso.

In questo caso il computer non andra' ad eseguire una ricerca di altri files.

COME ACCEDERE AI DATA FILES

Dopo aver aperto un file con un comando OPEN si puo' accedere a questo file sia per leggerci che per scriverci fino a quando il file stesso non sia chiuso con un comando CLOSE. Ricordiamoci ancora una volta che sia che si scriva sia che si legga, queste operazioni devono essere fatte in modo SEQUENZIALE. Cioe' il primo record scritto o letto sara' sempre il primo record del FILE, per cui se si vuole leggere il decimo record di un file sara' necessario prima leggere i primi nove. Nessun tasto della cassetta deve essere premuto prima che l' apposito messaggio non sia apparso sullo schermo. Attenzione alla posizione della cassetta. Infatti l' unita' a cassette iniziera' a scrivere immediatamente i dati non appena saranno premuti gli appositi tasti senza controllare ne che sul nastro sia scritto qulacosa ne che il nastro sia correttamente posizionato sull' inizio della superfice magnetica. Per scrivere dati su cassetta e' necessario usare il comando PRINT# che ha il seguente formato:

PRINT#f,data

dove f e' il numero di file logico che e' stato assegnato con un comando OPEN e che sara' riutilizzato con il comando CLOSE. Puo' avere un valore fra 1 e 255. Data saranno invece i dati da caricare sul file. Il comando PRINT# trasferisce i dati dalla memoria centrale del computer al buffer di cassetta. Quando il buffer e' stato completamente caricato nei suoi 191 Bytes di capacita' i dati sono scaricati su nastro appunto in BLOCCHI di 191 caratteri per volta.

IL DISCO

I dischetti possono immagazzinare sia files programmi che files di dati. A differenza che sulle cassette sui dischi si possono immagazzinare i files di dati in tre diversi modi, sempre sotto controllo di un programma:

FILES SEQUENZIALI
FILES RELATIVES
FILES RANDOM

La manipolazione di files su cassetta differisce in modo sostanziale da quella degli stessi files su disco per le seguenti ragioni:

1 - La velocita' di accesso ai dischetti e' molto piu' alta di quella delle cassette.

2 - Non esistono INIZI e FINE sulla superfice magnetica dei dischetti. Una unita' a dischi accede con facilita' a qualsiasi punto del disco cosa che ovviamente non avviene per la cassetta.

3 - La manipolazione di files di dati su cassetta o su disco differisce perche' la formattazione dei dati ed i metodi di accesso sono sostanzialmente diversi.

Non facciamoci ingannare dalla velocita' meccanica di rotazione del disco o della cassetta che se non e' uguale non determina comunque una grande differenza. La cassetta registra i dati in maniera sequenziale durante lo scorrimento del nastro e nello stesso modo li rilegge. Al contrario il disco immagazzina dati su un gran numero di tracce concentriche. La testina dell' unita' a dischi si sposta avanti ed indietro mentre il dischetto gira per trovare

il giusto punto in cui operare.
Per usare l' unita' a dischi non e' indispensabile sapere come le informazioni sono immagazzinate sulla superfice magnetica del supporto, tuttavia le conoscenze di questi argomenti renderanno piu' efficiente la programmazione. Per questo inizieremo la nostra discussione sui files disco descrivendo il modo in cui i dati sono immagazzinati sulla superfice magnetica del floppy.

COME IL DISCO IMMAGAZZINA I DATI

Un dischetto immagazzina i dati su un numero di tracce circolari concentriche. Queste tracce sono a loro volta divise in settori. I drives dell' unita' a dischi non scrivono dati lungo l' intera lunghezza della traccia, che e' utilizzata invece per la memorizzazione dei segnali di riferimento.

DIRECTORY DEL DISCO E BAM

Due tracce di ogni disco sono usate per l' indice del dischetto stesso. La prima o DIRECTORY TRACK contiene il nome che e' stato assegnato al dischetto, seguito dai nomi di tutti i files e dall' indirizzo di inizio del loro settore. La seconda traccia contiene la BAM o BLOCK AVAILABILITY MAP che identifica i blocchi dove sono o non sono allocati i files. Come abbiamo detto la BAM e' in pratica la rappresentazione della memoria disponibile su disco e della distribuzione degli spazi. Quando il sistema deve immagazzinare dati su disco, la BAM viene automaticamente collegata con il DOS per determinare quale spazio e' disponibile e quindi quanti blocchi possono essere salvati. Se

e' disponibile un spazio sufficiente per immagazzinare un dato file, allora l' operazione sara' coronata da successo e la BAM aggiornata per tener conto dello spazio utilizzato. Se invece il DOS riterra' che lo spazio non e' sufficiente allora verra' riportato un errore e l' operazione stessa di salvataggio non avra' effetto e verra' solo registrato il nome del programma nella Directory con un asterisco. A differenza di quanto accade sulla cassetta con una unita' a dischi si puo' andare direttamente all' inizio di un qualsiasi file sulla superfice del dischetto stesso, perche' ogni settore del disco e' egualmente accessibile. Per rendere possibile questo e' quindi necessario che ogni dischetto abbia un indice che contenga il nome di tutti i files ivi registrati e l'indirizzo del settore di partenza. Questo indice, simile quindi all' indice di un libro, e' appunto la DIRECTORY che fornisce anche il tipo di file che e' stato memorizzato e l' occupazione in blocchi di questo. Quando un file di dati su disco e' aperto, l' unita' prima di tutto leggera' la Directory dalla quale ottiene l' indirizzo del settore in cui ha inizio il file. Poi la testina di lettura/scrittura potra' posizionarsi direttamente all' inizio del file aperto. La Directory contiene le seguenti informazioni:

- -Nome del disco
- -Identificatore (ID) del disco
- -Numero di versione del DOS
- -Nome dei Files
- -Tipo dei Files
- -Numero dei blocchi usati
- -Puntatore al primo blocco dei Files
- -Numero dei blocchi disponibili

Vediamo ora come vengono trattati i records di un file su disco.

FILES RELATIVES

Tutti i records presenti in un file relative hanno la stessa lunghezza. Per questo e' facile calcolare l' indirizzo di settore per un singolo record di un file relative. Supponiamo di avere un file relative in cui i singoli records occupino mezzo settore. Cioe' che ne entri due per settore. Allora il decimo record di questo file relative sara' semplicemente rintracciabile sul quinto settore dall' inizio del file.

FILES SEQUENZIALI

I records di un file sequenziale possono avere differenti lunghezze. Per questo non si puo' calcolare il settore sul quale deve essere rintracciato un particolare record di un file sequenziale, appunto perche' la lunghezza del singolo record e' sconosciuta. La testina del dischetto puo' andare direttamente all' inizio di un file sequenziale, poiche' l' indirizzo del settore e' dato dalla Directory, ma una volta trovato questo inizio il file deve essere letto fino a quando non si trova il record desiderato. La ricerca e' quindi sequenziale e quindi simile a quella su nastro. Per trovare il decimo record di un file e' quindi necessario leggere i precedenti 9 records.

INDIRIZZAMENTO DEL DISCO

I settori assegnati su disco ad un file di dati non sono FISICAMENTE sequenziali sulla superfice del dischetto anche quando si utilizza un file di tipo SEQUENZIALE. Per esempio, quando si aggiungono records ad un file esistente, questi devono essere registrati senza andare a cadere

sul file successivo. Per questo il file dovra' essere proseguito, in casi di aggiunte, dovunque esistano settori liberi sulla superfice del dischetto. Il file si contrae quando si cancellano records per cui, con questa operazione si rendono disponibili nuovamente dei settori precedentemente allocati. Alla funzione di distribuzione del file sulla superfice del disco e' preposto il DOS cioe' Disk Operating System per cui la distribuzione su tutta la superfice del disco non presenta nessun problema quando si lavora con i files sequenziali. E' presente un puntatore in ogni settore che dice in pratica dove indirizzarsi per la successiva lettura o scrittura. In aggiunta alle operazioni di scrittura e lettura files su dischetto che vedremo separatamente e dettagliatamente per ogni tipo di accesso , il Basic della Commodore relativo a questo tipo di unita' consente le seguenti operazioni:

1-Preparazione di un nuovo dischetto.
2-Cancellazione di un disco vecchio e preparazione per un nuovo uso.
3-Visualizzazione della Directory del disco per vedere quali file sono immagazzinati, quanto spazio questi hanno occupato e quindi quanto ne resta utilizzabile.
4-Copia di un file
5-Copia di un intero dischetto
6-Cancellazione di un file o rimpiazzo dei files

PREPARAZIONE DI UN DISCO E INIZIALIZZAZIONE

A differenza di quanto avviene per la cassetta non si puo' prendere un dischetto vergine, inserirlo nel drive ed incominciare a scrivere i dati. Per prima cosa infatti la superfice

magnetica deve essere preparata ad accogliere i dati, i settori devono essere fissati e poi devono essere scritte la Directory e la BAM. Al dischetto deve essere assegnato un nome. Inoltre si puo' ripreparare per un nuovo uso un vecchio dischetto, naturalmente purche' sia in condizioni fisiche integre e sopratutto non sia rigato. Questa operazione cancella naturalmente tutti i dati vecchi, compresa la BAM e la Directory. Di norma la preparazione di un dischetto per il suo uso viene fatta in modo diretto, anche se questa routine puo' essere inserita in un menu' di programma. Per preparare un dischetto si deve per prima cosa eseguire un OPEN sul canale di comando. Poi si eseguira' un comando PRINT# usando il file logico specificato nella lista dei parametri del comando OPEN. Il comando PRINT# deve avere la seguente lista di caratteri, o parametri, racchiusa fra virgolette:

NEW o N

che identifica appunto l' operazione da eseguire.: (due punti) di separazione NOME DEL DISCO un nome qualsiasi che vogliamo dare , (virgola) anche questa di separazione XX identificatore del disco.
E quindi il formato generale del comando che segue il PRINT# sara':

"NEW:NOME DEL DISCO,XX"

NEW puo' essere rimpiazzato o abbreviato con la sola lettera N. Il nome del disco deve essere una stringa di lunghezza non superiore ai 16 caratteri.
XX deve essere un coppia di caratteri alfanumerici.
Nel comando OPEN con il quale si accede al

canale di comando si puo' mettere un qualsiasi numero di file logico, ma si deve specificare che l' unita' fisica e' la numero 8 e l' indirizzo secondario che deve essere il numero 15. Il comando NEW viene usato su un dischetto non FORMATTATO oppure su un dischetto che l' utente vuole riformattare e del quale quindi non interessano piu' i dati. Quando si usa il modo RIFORMATTAZIONE di un disco vecchio sara' cancellata la Directory preesistente e reinizializzata la BAM rendendo quindi nuovamente disponibili tutti i blocchi del dischetto.
In questo caso non dovremo specificare XX cioe' l' identificatore.
Vediamo qualche esempio.

```
OPEN 1,8,15
PRINT#1,"NEW:ESEMPIO,10"
```

RISULTATO: Viene aperto il canale di comando, formattato un disco che avra' per nome ESEMPIO e per identificatore 01.

INIZIALIZZAZIONE

Benche' non sia indispensabile questa funzione sul drive puo' accadere talvolta di doverla usare. Per inizializzare il dischetto e' necessario eseguire un OPEN sul canale di comando e successivamente un comando di PRINT# seguito dalle parole fra virgolette "INITIALIZE" o dalla letterea "I". Il comando "I" allinea la testina di lettura/scrittura con la traccia 1 del dischetto. I dischi sono normalmente inizializzati in modo programma e nessun dato sulla superfice del disco e' variata durante questa operazione che quindi puo' anche avvenire

con la finestrella coperta. Vediamo un esempio:

```
10 OPEN1,8,15

20 PRINT#1,"INITIALIZE" o   PRINT#1,"I"
```

Si puo' anche usare la forma abbreviata: OPEN15,8,15,"I" che si adopera in forma diretta quando si inizia ad operare su disco.

VALIDATE

Dopo che un dischetto e' stato usato per molto tempo, puo' succedere che la Directory debba essere riorganizzata. Infatti quando dati e programmi sono stati ripetutamente salvati (SAVE) e cancellati (SCRATCH), di queste operazioni possono esserci rimaste numerose tracce, in particolare in piccoli blocchi sparpagliati, appunto troppo piccoli perche' possano essere riutilizzati. In effetti la funzione di VALIDATE e' quella di cancellare tutti i files presenti nella DIRECTORY e di ricostruirne una nuova. Se durante questa operazione viene incontrato un errore allora la funzione di ricostruzione viene sospesa e si ritorna alle condizioni di partenza. Il comando VALIDATE riorganizzerra' allora il dischetto in modo tale che si possa disporre del massimo spazio effettivamente disponibile.
ATTENZIONE!!! C' e' un pericolo nell' uso di questo comando. Quando si usino i FILES RANDOM i blocchi ALLOCATI saranno DE-ALLOCATI con questo comando. Per questo motivo il VALIDATE non dovrebbe mai essere usato quando in un dischetto sono presenti i files random. Naturalmente e' un comando che si usa in forma diretta in una delle due forme:

PRINT#15, "VALIDATE" o PRINT#15,"V"

RENAME

Questo comando consente di cambiare nome ad un file di programmi o di dati. In effetti si tratta di una operazione molto veloce perche' l' unico cambiamento che avviene e' nella Directory del disco.
Sul disco naturalmente non deve esistere gia' un file con lo stesso nome utilizzato nel RENAME perche' in questo caso l' operazione non potra' avvenire ed avremo una segnalazione di errore: FILE EXISTS. Il formato di RENAME e':

PRINT#15,"RENAME0:vecchio nome=nuovo nome"

o nella forma abbreviata R al posto della lettera RENAME.

SCRATCH

Questo comando consente di cancellare files e programmi dal disco rendendo disponibili i blocchi per nuove informazioni. Si possono cancellare programmi uno alla volta o in gruppo come possiamo vedere dagli esempi. Il formato generale del programma e' il seguente:

PRINT#15,"SCRATCH0:nome del programma"

o abbreviando, S al posto della parola SCRATCH. Ammettiamo che siano presenti i seguenti files:

TEST,TRAIN,TRUCK,TAIL

possiamo usare:

```
PRINT#15,"S0:TR*"
```

se si desideri cancellare sia TRAIN che TRUCK. Usando invece:

```
PRINT#15,"S0:T*"
```

li cancelleremo tutti. Cancelleremo cioe' tutti i files che iniziano per T. Se per esempio la directory contenesse i files KNOW e GNAW usando: PRINT#15,"S0:?N?W" cancelleremo ambedue i programmi in quanto il ? sostituisce i caratteri ignoti all' inizio o nel mezzo del nome del file.

COPY

Questo comando consente, come del resto e' implicito nel nome, di effettuare una copia di un qualsiasi file di dati o programmi. Nel caso si disponga di un solo drive e' ovvio che la copia puo' essere fatta solo sullo stesso dischetto. Il formato di questo comando e':

```
PRINT#15,"COPY0:nuovo file=0:vecchio file"
```

oppure usando la lettera C al posto della parola COPY. Vediamo un esempio:

```
PRINT#15,"C0:MAILING
FILE=0:NOME,0:INDIRIZZO,0:TELEFONO"
```

Il comando COPY, in particolare per le unita' a singolo floppy presenta non pochi inconvenienti. Per questo e' stato messo a punto una procedura particolare (programma) che si trova in vendita con relativa facilita'.

COMANDI DISCO

Ricordiamo che questi comandi sono disponibili solo nella versione 128

DIRECTORY

Questo comando fara' apparire la Directory sullo schermo senza distruggere il contenuto della memoria. Si puo' usare il formato completo del comando:

DIRECTORY che ci dara' la Directory di ambedue i dischetti se ci sono oppure con il parametro D1 o D0. Oppure di puo' usare la forma abbreviatadiRd0 o diRd1. Ricordiamo che al posto del Directory puo' essere adoperata la parola CATALOG per intera o con l' abbreviazione: CaD0 o CaD1

COLLECT

Questo comando ha la stessa funzione del VALIDATE. Le operazioni a cui da luogo sono le seguenti:
-Ricrea la mappa di disponibilita' dei blocchi in accordo con i dati validi su disco.
-Cancella dalla Directory i files che non sono stati chiusi in modo corretto.
Il formato dell' istruzione COLLECT e':

COLLECT Dx

DUPLICATE

Questo comando esegue prima di tutto la

formattazione del dischetto di destinazione e poi trasferisce ciascun blocco di informazioni dal dischetto sorgente al dischetto destinazione, creando una copia esatta del disco sorgente. Poiche' il DOS 1, DOS 2, ed il DOS 2.5 usano protocolli di formattazione diversi questo comando non puo' essere usato indifferentemente con dischetti preparati su drives diversi.

```
PRINT#1,"DUPLICATE ddr=sdr"
```

oppure con la notazione abbreviata:

```
PRINT#1,"Dddr=sdr"
```

ddr= dischetto di destinazione
sdr= dischetto sorgente.

E' molto importante non rovesciare l' ordine in cui viene dato il numero di drive perche' in questo caso si andrebbe incontro ad una perdita dei dati e tale errore non sarebbe rimediabile in alcun modo. Sara' quindi buona norma proteggere con l' apposita etichetta sulla finestra il disco sorgente.

BACKUP

Il comando BACKUP esegue le stesse funzioni del comando visto in precedenza. Il suo formato e':

```
BACKUP Dsdr TO Dddr
```

dove sdr e ddr hanno gli stessi significati visti per il DUPLICATE.
E' importante notare che il formato dell' istruzione BACKUP differisce dal formato dell' istruzione DUPLICATE nel senso che l' ordine dei drive e' rovesciata.

CONCAT

Questo comando permette all' utente di concatenare files. Il formato di questa istruzione e':

```
CONCAT Dsdr,"sfn" TO Dddr, "dfn"
```

A fine operazione il file chiamato dfn sul drive ddr conterra' sia il contenuto del file dfn che del file sfn.Esempio:

```
CONCAT DO,"ALFA" TO D1,"BETA"
```

Dara' come risultato in BETA del drive 1 il contenuto in dati di BETA e ALFA.
ALFA restera' inalterato.

COMANDI PER LA MANIPOLAZIONEDI DATI DISCO

I seguenti comandi consentono all' utente di operare su disco per comunicare alla periferica un qualsiasi tipo di dati e per poi riutilizzarli. In tutte le versioni del DOS cioe' del Disk Operating System (Sistema Operativo su Disco) sono disponibili i seguenti comandi:

```
OPEN lfn,8,sa,"dr:fn"
CLOSElfn
LOAD "dr:fn",8
SAVE "dr:fn",8
VERIFY"dr:fn",8
PRINT #
GET #
INPUT #
```

I seguenti comandi sono invece disponibili solo sulla versione 128

```
DOPEN #lfn,"fn"
DCLOSE #lfn
DLOAD "fn"
DSAVE "fn"
RECORD #lfn,R,B
```

Dove:

lfn = numero di file logico
fn = nome del file
dr = numero del drive
sa = indirizzo secondario
lf = file logico

SAVE e DSAVE

Questo comando consente di copiare un programma dalla memoria del computer al dischetto per il suo immagazzinamento.
Cio' puo' essere ottenuto con SAVE in tutti i tipi di Basic e con DSAVE quandi si usa la versione 128. Ogni dato trasferito con i comandi SAVE e DSAVE e' automaticamente definito e quindi registrato dal Disk Operating System come file programma e viene registrato nella Directory del disco con la lunghezza in blocchi, il nome, e l' indicatore PRG. Quindi entrambi i comandi trasferiscono dati in forma di programmi dalla memoria del computer ad un dato dischetto. Sara' necessario specificare il numero del drive, il nome del programma ed il numero di periferica. Il formato del comando SAVE e':

```
SAVE "dr:fn",dn
```

Ricordiamo che il nome del file per il disco e' obbligatorio a differenza di quanto accade per la cassetta e che sono contati, come numero di caratteri anche gli spazi bianchi. Il comando DSAVE esegue le stesse funzioni viste in precedenza solo che si da in una forma diversa e

piu' semplice.Il formato e':

DSAVE"fn"Ddr

LOAD e DLOAD

Un programma o comunque una serie di dati immagazzinati su dischetto come programma puo' essere riletto con uno dei comandi LOAD (per tutti) o DLOAD.I comandi LOAD e DLOAD trasferiscono quindi un file PRG da un dato dischetto nella memoria interna RAM del computer. Si deve specificare il numero del drive, il nome del programma ed il numero della periferica.
Il formato del comando LOAD e':

LOAD "dr:fn",dn

Dove per i parametri valgono le regole precedentemente esposte tranne per il fatto che il nome del file (parametro fn) in questo caso dovra' essere un programma salvato precedentemente con un comando SAVE.Il comando DLOAD esegue le stesse funzioni solo che deve essere dato in un' altra forma:

DLOAD"fn",Ddr

La corretta esecuzione di un comando LOAD o DLOAD porta anche alla chiusura di tutti i files aperti. Per questo sara' necessario un nuovo comando di OPEN per poter continuare a comunicare con il disco sia per l' invio di comandi che per ripristinare il canale di errore.Ricordiamo inoltre che questi comandi portano alla distruzione del programma eventualmente presente in precedenza nella memoria del computer.

VERIFY

Il formato di questo comando e':

VERIFY"dr:fn",8

Questo comando verifica che un programma, il cui nome e' specificato nel parametro "fn", immagazzinato su floppy disk contenga le stesse informazioni di un programma presente nella memoria del computer. E' lo stesso di quello visto in precedenza per la cassetta.

OPEN

Come e' stato gia' spiegato in precedenza e piu' volte questo comando fissa una corrispondenza univoca tra un numero di file logico ed un file esistente su disco.Questo comando riserva anche uno spazio nel buffer dell' unita' a disco per le operazioni sul file che deve essere aperto. Il formato completo del comando OPEN e' il seguente:

OPENlfn,dn,sa,"dr:fn,ft,modo"

lfn = e' il numero di file logico
dn = e' il numero della periferica e in questo caso normalmente il numero 8.
sa = e' l' indirizzo secondario che puo' essere un numero qualsiasi fra 2 e 14 e deve essere usato sia per l' input che per l' output dei dati come viene specificato nel modo. (vedi anche la nota seguente).
dr = numero del drive che potra' essere 0 o 1 nelle unita' a doppio drive.
fn = nome del file.
ft = tipo del file che potra' essere :

SEQ = Sequenziale
USR = Utente
REL = Relative
PRG = Programma

modo = che descrive in che modo il canale di trasferimento dati debba essere utilizzato. Potra' essere quindi una delle opzioni:

READ (R) per la lettura
WRITE (W) per la scrittura.

DOPEN

Questo comando e' disponibile solo sulla versione 128. Quando e' utilizzato deve essere impiegato per creare files relatives o sequenziali di lunghezza fissa. Il formato di DOPEN e':

```
DOPEN#lfn,"fn",Ddr,Lrl,(ON Udn)(,W)
```

dove:

lfn,fn,dr sono gli stessi parametri spiegati in precedenza per il comando OPEN.
Lrl = definisce la lunghezza del record che deve appunto essere uguale a rl
ON Udn = specifica il numero di periferica e che se manca sara' automaticamente posto uguale a 8.
W deve essere specificato per consentire il modo scrittura. Se W non e' specificato per i files sequenziali, questi verranno aperti per una operazione di lettura.

CLOSE e DCLOSE

Questo comando serve per chiudere un file precedentemente aperto con un comando OPEN. Il

suo formato e' il seguente:

CLOSE lfn

Ricordarsi di chiudere sempre un file dopo aver terminato di lavorarci. Non e' infatti possibile avere piu' di 10 file contemporaneamente aperti sul computer e piu' di 5 sul disco. Per questo e' buona norma prendere l' abitudine di chiudere un file al piu' presto possibile perche' questo metodo ci consentira' di avere una riserva da poter opportunamente impiegare quando se ne manifesti la necessita' e quindi di avere anche il massimo numero dispobile di files.
La sintassi del comando DCLOSE e' la seguente:

DCLOSE #lfn

lfn = e' il file che deve essere chiuso.Il comando DCLOSE puo' essere impiegato anche nella seguente maniera e formato:

DCLOSE ON Udn

dove:

dn = e' il numero di periferica dell' unita' a dischi (di norma ed in mancanza di indicazioni e' 8).
Quando e' utilizzato in questa forma il comando DCLOSE chiudera' tutti i files attivi sull' unita' specificata.

PRINT #

Questo comando, gia' visto a proposito della cassetta, serve ad inviare una stringa di comando alla periferica disco. Il formato e':

PRINT#lfn,"stringa di comando"

lfn = numero di file logico preventivamente aperto utilizzando l' indirizzo secondario 15.
stringa di comando = e' appunto un comando di manipolazione di un file disco o semplicemente un comando al disco. Questi comandi sono stati gia' visti in precedenza e saranno anche approfonditi in seguito nella parte relativa alla trattazione dei vari tipi di files.
PRINT# deve anche essere usato per trasmettere dati ad un file sequenziale o relative preventivamente aperto. Deve essere utilizzato un punto e virgola (;) come carattere terminatore per ogni comando PRINT# quando si stia utilizzando computer provvisti di BASIC 3.0 per evitare di inviare LINE FEEDS estranei al dischetto. Questi caratteri sono scritti sul dischetto come parte del TERMINE DATI dalla routine BASIC PRINT#. E' importante essere a conoscenza di questo aspetto perche' il ritorno carrello da solo e' visto come CARATTERE DI TERMINE dal DOS. Il LINE FEED e' allora immagazzinato nel file come primo carattere del successivo record. Per evitare cio' utilizzate il seguente formato:

PRINT#2,"PIPPO";CHR$(13)

Il CHR$(13) e' il ritorno carrello necessario per un' appropriato termine del record sul disco.
Piu' variabili possono essere scritte su disco allo stesso tempo. Vediamo un esempi.

PRINT#lfn,A$,B$,C$

che portera' come risultato la scrittura di una variabile composta dalla somma delle singole variabili e cioe':

A$+B$+C$

e che percio' potra' essere ricercata e riletta come unica variabile.

INPUT #

Il comando INPUT# e' utilizzato per trasferire informazioni da una periferica come appunto un disco nella memoria del computer.
INPUT # e' valido solo quando e' utilizzato in un programma e solo quando si riferisce ad un file logico che sia stato aperto (con OPEN) per l' ingresso dati. Il formato di INPUT# e':

INPUT#lfn,A$ o INPUT#lfn,A

lfn = e' un file preventivamente aperto utilizzando l' indirizzo secondario 15.
A$ = e' una normale variabile stringa che contiene i dati da trasferire.
A = e' una normale variabile numerica che contiene i dati da trasferire.
INPUT# puo' essere utilizzata per trasferire piu' stringhe allo stesso tempo.

INPUT#lfn,A$,B$,C$

A$,B$ e C$ conterranno i dati che devono essere riletti dal computer.
Nell' esempio qui sopra e' necessario ricordarsi che i dati devono essere stati scritti con un CHR$(13) perche' se non troveranno il carattere di ritorno carrello come separatore, allora avremo la rilettura di una stringa intera o di una somma di stringhe. Se si vogliono cioe' trovare tante stringhe separate si devono anche scrivere separatamente. Ricordiamo che nessuna

stringa puo' contenere piu' di 80 caratteri in fase di INPUT. Per stringhe piu' lunghe di 80 caratteri e' necessario utilizzare il comando GET#.

GET#

Il comando GET# ha la stessa funzione del comando INPUT# visto in precedenza solo che opera su un Byte di dati per volta. Anche questo comando non puo' essere utilizzato che in modo programma, e quindi come per il GET normale, non e' possibile adoperarlo in modo diretto, e solo quando e' riferito ad un file che sia gia' stato aperto. Il formato e':

```
GET#lfn,A$
```

GET# deve anche essere usato per trasferire piu' bytes di informazioni o di dati ed e' quindi utile per ricercare stringhe che sono state scritte in precedenza su disco in un formato non accettabile per un comando di INPUT. Cioe' normalmente per stringhe superiori a 80 caratteri.Vediamo un esempio commentato.

```
10AA$=""
20FORI=1TO254
30GET#lfn,A$
40AA$=AA$+A$
50NEXT
```

Con questo programma sara' possibile trasferire da disco all' unita' centrale una stringa della lunghezza di 254 caratteri ed immetterne il contenuto in memoria, dell' unita' centrale, nella variabile AA$.

RECORD#

Il comando RECORD# e' utilizzato prima di comandi INPUT#, PRINT# o GET# per posizionare il puntatore del file (FILE POINTER) ad un desiderato record di un file relative. Per esempio se il puntatore del record e' fissato al di la' dell' ultimo record e viene utilizzato il PRINT #, viene generato il giusto numero di records per espandere il file al record richiesto. Ricordiamo che questo comando, come del resto la gestione di file relatives sono disponibili solo per 128. Il formato del comando e':

```
RECORD # lfn,r,b
```

lfn = e' il numero di file logico preventivamente aperto con un comando DOPEN.
r = e' il desiderato numero di record.
Questo parametro puo' essere sia un nome di variabile che un valore. Se e' un nome deve essere racchiuso fra parentesi, mentre se e' un valore deve essere compreso in un' intervallo fra 0 e 65535.
b = e' la posizione del byte richiesto entro il record. La posizione del byte e' opzionale. b deve essere compreso fra 1 e 254. Il seguente esempio illustra come il comando RECORD e' utilizzato con un INPUT#. Esempio

```
10RECORD #1,120
20INPUT#1,A$
```

Dove nella linea 10 si usa il comando RECORD per selezionare il record interessato. Nella linea 20 si esegue l' input del prossimo gruppo di dati come stringa e lo assegna ad una variabile A$.

CAPITOLO SESTO

IL LINGUAGGIO MACCHINA ED IL SISTEMA OPERATIVO

Una delle cose che maggiormente scoraggiano il normale utente del computer ad andare oltre il piu' o meno completo apprendimento del Basic e' il numero delle nuove parole, e piu' in generale dei termini da imparare. Inoltre molti scrittori, in particolar modo quando trattano del Linguaggio Macchina, partono dal concetto che il potenziale utente abbia una conoscenza di elettronica non superficiale e danno quindi per scontato una serie di concetti e di parole.
Come abbiamo gia' detto tutto cio' che noi pensiamo invece e' che l' utente che si appresta a leggere questo pagine abbia assimilato un po' del Basic visto in precedenza.
Gran parte di quanto scriviamo in questo capitolo e' tratto da precedenti manuali EVM fra i quali il piu' importante e' il CORSO DI ASSEMBLER per CBM64 al quale rimandiamo per approfondimenti.
Per iniziare parliamo della memoria del computer e dei concetti, fondamentali che ad essa si legano. Una unita' di memoria, in un computer e' come un circuito elettrico che agisce con un interruttore allo stesso modo di quando si entra in una stanza e si accende la luce. Noi sappiamo che un interruttore puo' essere girato in un senso o nell' altro, ON o OFF, acceso o spento, cioe' puo' avere due posizioni e che resta in quella posizione fin quando non si esegue una operazione che appunto lo sposti da una posizione ad un' altra. Una unita' di memoria

cosi' concepita e che e' l' unita' base del nostro sistema si chiama BIT che e' l' abbreviazione di BINARY DIGIT.
Supponiamo di voler effettuare delle segnalazioni disponendo solo di una lampadina e di un interuttore. Quando l' interruttore e' ON, in altre parole girato per un certo verso e da ora in poi intendiamo con questo concetto acceso , assumeremo di voler dire di SI. Al contrario quando e' OFF, cioe' spento, di dire di NO. Possiamo dire che abbiamo due, e due sole, condizioni possibili o STATI della luce o del segnale che vogliamo mandare.
Vediamo ora che cosa puo' accadere quando le luci e quindi gli interruttori da uno diventano due. Abbiamo quattro differenti possibili combinazioni: a)entrambi OFF; b) A ON e B OFF; c) A OFF e B ON; d) entrambi ON.
Possiamo ora affermare che essendo 4 combinazioni si possono inviare quattro messaggi differenti. Con una linea quindi 2 codici, con due linee 4 codici cioe' 2x1 e 2x2.
Con tre linee sara' 2x2x2 cioe' 8, con 4 linee e quindi quattro interruttori e quattro lampadine 2x2x2x2 cioe' 16 e cosi' via. In altre parole possiamo dedurre una formula che e' 2 elevato al numero delle linee. Con 8 linee avremo 2 all' ottava cioe' 256 combinazioni. Dobbiamo ora vedere come impiegare questi concetti.
Un sistema particolarmente utile e che noi impiegheremo e' quello del CODICE BINARIO o BINARY CODE che e' un sistema di numerazione che impiega solo due valori 0 e 1.
Possiamo pensare allo ZERO (0) come interruttore spento o OFF e all' UNO (1) o condizione 1 come interruttore acceso o ON. In questo modo, utilizzando solo 8 interruttori potremo impiegare 256 combinazioni di segnali diverse. Questo raggruppamento di 8 bit e' chiamato BYTE ed e' diventato un' unita' di misura standard

che fra l' altro viene impiegata per dare la grandezza della memoria del computer.
Dopo aver visto i concetti di BIT e i BYTES possiamo affermare che la memoria di un computer altro non e' che una serie di interruttori.
Osserviamo ora che nel computer ci sono due tipi di memorie, una permanente e quindi, per riportarsi all' esempio iniziale, con tutta la serie di interruttori che sono gia' fissati in una determinata posizione e di conseguenza i segnali sono codificati al momento stesso della costruzione della memoria. Questa si chiama ROM cioe' READ ONLY MEMORY o memoria a sola lettura.
Normalmente questo tipo di memoria contiene il Sistema Operativo, l' interprete Basic e tutto cio' che consente al computer di comunicare con il mondo esterno o, come meglio si dice di interagire con esso.
Quando scrivete un programma invece il computer lo immagazzina in altra parte della memoria, sempre in forma numerica, e che puo' essere utilizzata piu' volte come lettura e scrittura.
Questo diverso tipo di memoria puo' quindi essere LETTO e SCRITTO e si chiama RAM o RANDOM ACCESS MEMORY.
Gli interruttori posizionati in certo modo o meglio le informazioni scritte in questo tipo di memoria possono come abbiamo gia' detto essere cambiate quante volte si vuole, ma quando si toglie tensione al circuito, o in altre parole si spegne la macchina, queste informazioni vengono irrimediabilmente perse.

IL MICROPROCESSORE

Il terzo blocco fondamentale del computer e' la CPU che sta per CENTRAL PROCESSING UNIT ed ha una particolare importanza. Infatti questa unita' puo' essere definita come la parte piu'

importante del sistema o UNITA' OPERATIVA. Questa e' costituita da un solo integrato che nel nostro caso e' il microprocessore 8502, cioe' un piccolo pezzo di plastica all' interno del quale e' presente un pezzo di silicio sul quale sono stati riportati con processi di microfotografia una serie di circuiti. Da questo pezzetto di plastica escono delle connessioni o PIN in numero pari per lato.
Cosa fa la CPU?
Praticamente tutto ed inoltre le azioni che puo' eseguire sono straordinariamente semplici e poche. E' infatti la combinazione delle stesse azioni che da la capacita' al computer di eseguire calcoli complessi. La CPU puo' caricare (LOAD) un Byte. In altre parole un Byte in un certo punto della memoria puo' essere immesso all' interno della CPU. Puo' eseguire anche l' azione opposta cioe' il trasferimento (STORE) in qualsiasi punto, consentito, della memoria. Queste due azioni sono quelle in cui la CPU impiega la maggior parte del suo tempo operativo e vi renderete conto di questa affermazione continuando nello studio del corso.
Per rendersi conto di questa azione in apparenza semplice, vediamo un caso pratico e cioe' cosa accade quando si preme un tasto, ad esempio la lettera G.
La CPU tratta la tastiera come una parte di memoria e il video come se fosse un altra parte e copia quindi, trasferendo, da una parte all' altra. Naturalmente questo e', come vedremo in seguito,, una semplificazione notevole del concetto ma dovrebbe chiarire l' operazione.
Altre funzioni che esegue la CPU sono quelle aritmetiche spesso solo addizioni e sottrazioni su numeri da 0 a 255.
Quando deve eseguire operazioni piu' complesse come logaritmi, moltiplicazioni, potenze e su numeri piu' grandi vedremo che verranno

utilizzate le informazioni, in questo caso, veri pezzi di programma, presenti in ROM.
Oltre questo la CPU esegue anche operazioni LOGICHE che consistono nel confrontare il contenuto dei bits di due Bytes e danno un risultato che dipende dal contenuto dei singoli bits e dall' operazione logica che si esegue.
Un' altra serie di azioni e' quella dei salti o JUMP. Un salto consente un cambio di indirizzo allo stesso modo dell' azione del GOTO nel Basic e quindi da la possibilita' di eseguire dei controlli e di prendere delle decisioni.
Naturalmente tutte queste informazioni viaggiano tramite segnali elettrici in cui sono preventivamente trasformati e, come possiamo osservare dalla tavola del diagramma a blocchi, collegati tramite un BUS o collegamento. Sotto un altro punto di vista possiamo guardare alla CPU anche come una periferica collegata appunto agli altri componenti del sistema tramite il Bus.
Di identica importanza e' il fatto che lo stesso microprocessore possa essere programmato inviandogli segnali elettrici.Questi segnali sono inviati a 8 PINS o piedini chiamati DATA PINS, e per questo non sara' molto difficile comprendere che questi 8 PINS corrispondono agli 8 bits di un Byte. Ogni Byte della memoria puo' quindi essere connesso con la CPU tramite segnali elettrici.

I SISTEMI NUMERICI

Come abbiamo gia' detto il microprocessore non comprende altro che il modo di numerazione BINARIO. Allo stesso modo che il sistema decimale opera sulle cifre 1,2,3,4,5,6,7,8,9,0 il binario opera su 0 e 1. Per questo si dice che il decimale ed il binario sono dei sistemi

di numerazione che operano rispettivamente in base 10 e 2.
Prendiamo per esempio il numero 3783. Questi e' composto da migliaia (3), da centinaia (7), da decine (8) e da unita' (3). Per cui si puo' scrivere:

3783 = (3x1000) + (7x100) + (8x10) + (3)

o meglio ancora

3783 = (3^10) + (7^10) + (8^10) + (3^10)

Se noi lo inseriamo in un sistema a base 2 avremo:

3873 = (1x2048) + (1x1024) + (1x512) + (1x256) + (1x128) + (1x64) + (0x32) + (0x16) + (0x8) + (1x4) + (1x2) + (1x1)

oppure con la forma esponenziale come visto in precedenza

3873 = (1x2^11) + (1x2^10) + (1x2^9) + (0x2^8) + (1x2^7) + (1x2^6) + (0x2^5) + (0x2^4) + (0x2^3) + (1x2^2) + (1x2^1) + (1x2^0)

Il numero 3783 in base 2 si scrivera' dunque:

3783 = 111011000111

Possiamo concludere che il numero detto puo' essere rappresentato in binario con 12 bits. In generale e' valida la regola che N bits permettono di codificare 2^n numeri che sono compresi fra 0 e 2^n-1. I codici operativi del 8502 sono dati in forma di Byte che e' l' insieme di 8 Bits. Percio' possono esistere con il 6510 un massimo di 2^8 = 256 codici operativi, anche se in effetti sono di meno.

Per quale motivo e' stato scelto il codice binario dato che noi siamo abituati a contare in base 10?
Il motivo e' nella facilita' di messa in opera dal punto di vista delllla costruzione del circuito Hardware. Infatti e' molto piu' semplice realizzare e costruire dei circuiti che possono avere 2 differenti stati, come visto in precedenza (0 o 1, 0 Volts o 5 Volts, SI o NO) invece che realizzare dei circuiti che possono avere 10 differenti stati.

L' ESADECIMALE

Abbiamo visto il sistema binario, che se e' facile da realizzare dal punto di vista della circuiteria elettronica non lo e' altrettanto dal punto di vista dell' interazione umana. E' stato messo a punto quindi un sistema ulteriore di numerazione che vedremo facilitera' molto le operazionni sui Bytes. Allo stesso modo che il sistema decimale comporta l' impiego di 10 cifre cosi' l' esadecimale ne comporta l' impiego di 16:

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

di cui mostriamo gli equivalenti:

ESA	DEC.	BINARIO
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111

8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Vediamo ora come sia possibile convertire un numero binario in esadecimale e viceversa. E' molto semplice: basta dividere il numero esadecimale in tante cifre e sostituirle con i 4 valori corrispondenti in binario visti nella tabella precedente. L' esadecimale 69 sara' quindi:

6 = 0110   9 = 1001

quindi;

$69 = % 01101001

ed ancora:

$326D sara'

3 = 0011 2 = 0010 6 = 0110 D = 1101

$326D = % 0011001001101101

D' altra parte per eseguire l' operazione opposta sara' sufficiente spezzare il numero binario in tanti gruppi di quattro cifre partendo da sinistra.

11111010 sara' FA

IL CODICE ASCII

Abbiamo visto diversi modi di codifica dei numeri, altri ne vedremo in seguito come il BCD e l' ottale , ma e' ora necessario sapere in che modo codificare i caratteri alfanumerici. Anche per questo motivo e' stato messo a punto il codice ASCII, acronimo di AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE.
Si tratta di un codice di 8 bits in cui pero' il bit piu' significativo e' utilizzato per il controllo degli errori. Si tratta del bit di parita'.
L' informazione sara' quindi contenuta in 7 bits cosa che per le regole esposte innanzi permette di codificare 128 caratteri differenti.
Questi 128 codici ci permettono di rappresentare tutte le lettere dell' alfabeto, le cifre oltre ad una serie di caratteri speciali e di controllo.

IL LINGUAGGIO MACCHINA

Perche' utilizzare il linguaggio macchina?
Quando si acquista un HOME o un PERSONAL computer come il C128, normalmente si ha disponibile il linguaggio Basic al momento dell' accensione. Con il Basic si possono risolvere la maggior parte dei problemi abbordabili con questo tipo di computer e non e' eccessivamente difficile da adoperare. Perche' allora imparare un' altro linguaggio? Quali vantaggi da realmente questo nuovo linguaggio di programmazione, che ci auguriamo di poter spiegare bene in questo corso, sul Basic da ripagare i nostri sforzi?
Il vostro C128 ha inserito, e lo vedete al momento dell' accensione scritto sullo schermo, il linguaggio Basic al suo interno, ma non lo

puo' comprendere, almeno direttamente. Infatti quando esegue un comando Basic che magari avete inserito da tastiera, questi, tramite il Sistema Operativo che contiene anche l' interprete Basic, viene convertito in una serie di istruzioni in Linguaggio Macchina, queste eseguibili direttamente dal sistema. Per eseguire un semplice comando come POKE 1024,10 l' interprete deve fare un mucchio di lavori che portano via tempo. Vediamoli per comprendere meglio.
Per prima cosa l' interprete cerca e controlla che le prime due parole del comando siano presenti nella TAVOLA delle parole riservate. Poi osserva che a questo comando sono necessari 2 argomenti, legge il primo (1024) e lo converte in binario. Ricordiamoci infatti che il computer lavora in binario.
Cerca quindi e trova il secondo argomento (10) e lo converte in binario. Infine scrive questo secondo valore nella locazione di memoria indicata dal primo argomento. Questa serie di operazioni richiede 2 millesimi di secondo. La stessa istruzione potrebbe essere scritta in assembler:

```
LDA #10
STA  1024
```

L' esecuzione di queste istruzioni richiede 6 milionesimi di secondo che equivale a meno di un trecentesimo del tempo del Basic.
Alcune operazioni come SORT (ordinamenti) e calcoli matematici impiegano realmente molto tempo e se si utilizzano grandi gruppi di dati possono rendersi necessarie anche molte ore con il Basic.
Altre operazioni poi non possono essere eseguite con il Basic come ad esempio la funzione di INTERRUPT.

Un' altro importante vantaggio e' costituito dall' utilizzo della memoria. Infatti un programma in Linguaggio Macchina ben scritto sara' almeno 10 volte piu' corto dell' equivalente in Basic. Stesso discorso vale per l' occupazione della memoria. Infatti il Basic richiede 2 Bytes per rappresentare un valore intero compreso fra 0 e 255. In Linguaggio macchina sara' necessario solo 1 Byte.
Tutto questo ci consente di affermare con sicurezza che, anche se presenta una certa difficolta' ad essere appreso, comunque non di molto superiore al Basic, vale veramente la pena di spendere questo tempo. La funzione di tutti gli Assembler e' quindi quella di tradurre un programma scritto in codici mnemonici, e quindi semplici da ricordare, in Codice Macchina.

L' ACCUMULATORE

Il cuore del microprocessore 8502 e' un registro chiamato ACCUMULATORE ed abbreviato con la lettera A. Si tratta di un registro ad 8 bit attraverso il quale passeremo quasi sempre e che quindi puo' immagazzinare numeri da 0 a 255.
Le istruzioni del 8502 consentono di scrivere direttamente entro questo registro.

LE LABEL

Sono quei nomi da mettere prima dei codici mnemonici e che servono all' interno del programma come riferimento per i vari salti.
Sempre con riferimento al Basic possiamo considerare le LABEL come i nomi delle Subroutines solo che in questo caso, invece di utilizzare un indirizzo si utilizza un nome al quale corrisponde un indirizzo.

I REGISTRI INDICE

Oltre all' Accumulatore, il 8502 ha due registri sempre di un solo Byte, detti REGISTRI INDICE:

REGISTRO INDICE X

REGISTRO INDICE Y

Ognuno di questi che d' ora in poi chiameremo semplicemente registro X o Y ha, come l' Accumulatore, la possibilita' di immagazzinare valori nell' intervallo da 0 a 255 essendo registro da 8 bits.

NOTA

Ricordiamo ancora una volta che un registro e' una locazione di memoria nella quale puo' essere caricato un valore. Questo valore di norma e' compreso, come abbiamo detto in un' intervallo fra 0 e 255 per i registri da 8 bits e fra 0 e 65535 per i registri da 16 bits. Per il momento inoltre i registri X e Y sono mostrati con funzionamento simile fra loro sebbene in effetti differiscano come comportamento. Il grande vantaggio di questi registri indice e' che il valore in essi contenuto puo' essere incrementato o decrementato, di 1 per volta, in modo semplice.
Naturalmente non sono solo questi i vantaggi e le possibilita' che essi hanno. Altro punto da prendere in considerazione e' l' ALU o ARITHMETIC AND LOGIC UNIT cioe' unita' aritmetico-logica che si trova all' interno del microprocessore stesso ed e' da questi usata appunto per tutte le operazioni aritmetico logiche.
La ALU ha due ingressi per i dati sui quali esegue le operazioni ed un' uscita tramite la

quale i risultati delle operazioni stesse vengono inviati all' Accumulatore . La strada sulla quale i dati passano si chiama:

DATA BUS

I SALTI ED IL PROGRAM COUNTER

Come del resto avviene per il Basic, nessun programma procede attraverso una serie di passi ininterrotti, senza salti a subroutines, a Kernal routines o altro.
Sara' dunque necessario un registro che tenga conto di questo lavoro. Questo e' un registro a 16 bit che contiene gli indirizzi del prossimo comando da eseguire.
In realta' si tratta di due memorie di 8 bit ciascuna inserite entro l'8502. Non appena il PC avanza, ogni Byte di memoria si incrementa di uno in modo tale da puntare alla successiva locazione di memoria che conterra' quindi i dati richiesti. L' avanzamento del PC e' sequenziale per cui ad ogni incremento unitario, corrisponde un puntamento sulla locazione di memoria immediatamente successiva.

SALTI INCONDIZIONATI

Sono comandi che dicono al programma di saltare ad un certo indirizzo, semplicemente, cioe' senza condizioni.
Sul 8502 esistono solo due istruzioni di questo tipo. La prima e':

JMP JuMP to the specified address

Cioe' salta ad un dato indirizzo
Per esempio, JMP 834 ordina di saltare alla

locazione di memoria 834. Questo potrebbe, e spesso e' cosi', essere un modo di inserire pezzi di programma dimenticati o parti di programma in aggiunta.

SALTI CONDIZIONATI

Abbiamo visto prima dei salti incondizionati, ma un qualsiasi programma che necessiti di un minimo di controllo avra' la necessita' di SALTI CONDIZIONATI. Per fare un' analogia con il Basic possiamo prendere il comando di condizionamento IF....THEN:

```
10 IF X=Y THEN 500
```

In questa linea i valori X e Y, che sono stati immagazzinati in memoria sono confrontati fra loro e se si verifica la condizione di eguaglianza, almeno in questo caso, si salta alla linea specificata dopo il THEN. Il 8502 puo' eseguire questo tipo di operazione in una molteplicita' di modi.
Uno di questi e' attraverso l' uso di un particolare registro chiamato REGISTRO DI STATO o STATUS REGISTER (SR) o anche conosciuto come PROCESSOR STATUS WORD.
Lo STATUS REGISTER e' un registro di 8 bits come l' Accumulatore, i registri X e Y ma viene usato in maniera differente . Mentre gli altri registri sono usati per immagazzinare e manipolare Bytes , questo registro divide e quindi considera separatamente i sui singoli BITS come FLAGS o segnali. Di norma il 8502 manipola uno solo di questi Flags per volta, sia fissandone il suo valore a 0 oppure a 1 sia controllando se il valore e' a 0 oppure a 1. In altre parole su questi Flags (che sono poi i singoli bits del SR) si puo' , di volta in volta

scrivere o leggere il valore relativo. Un esempio di uno di questi Flags e' il flag Z o flag ZERO.
Quando viene eseguito un programma o una manipolazione di dati che produca un risultato di 0 (zero) in un determinato registro (A,X o Y) allora il flag Z viene messo a 1. Se invece il risultato e' diverso da 0 allora lo Z flag viene messo a 0. Si puo' quindi dire che l' indirizzamento e' una modifica del comando fatta per cambiare la sua funzione in modo particolare. L' indirizzamento fa si che il 8502 punti (o sia puntato o indirizzi) ad una locazione di memoria sia direttamente che indirettamente. La strada seguita dipende dal modo particolare di indirizzamento usato.
Gli indirizzamenti sono uniformi attraverso tutta la memoria disponibile tranne che per i primi 256 Bytes di memoria (dalla locazione 0 alla 255).
Per indirizzare queste locazioni (o prima pagina di memoria) e' necessario solo 1 Byte mentre tutte le altre pagine necessitano di 2 Bytes.

NOTA

Ricordiamo che l' intera mappa di memoria del C128 puo' essere divisa in pagine ognuna delle quali di 256 Bytes e che la prima pagina e' chiamata appunto PAGINA ZERO che ha uno speciale modo di indirizzamento che vedremo fra poco.
Ricordiamo inoltre che quando in risposta ad un comando si salta da una pagina all' altra a livello di temporizzazione verra' usato un ciclo addizionale.
Vediamo una breve sintesi sugli indirizzamenti

INDIRIZZAMENTO IMMEDIATO

Questo modo di indirizzamento consente che un numero sia caricato immediatamente entro un registro o per essere usato direttamente come termine di un confronto. Questo modo di indirizzamento e' identificato chiaramente dal segno POUND (#) che pprecede il valore che deve essere caricato.
LDA #10 e' un modo immediato di indirizzare l' Accumulatore e consente che il valore immediatamente seguente sia caricato nell' accumulatore stesso. La corrispondente istruzione Basic potrebbe essere A=10.
Questo modo e' utilizzato per caricare un registro con una costante e puo' lavorare anche con i registri X e Y.
Quando si impiega questo modo di indirizzamento il valore che deve essere caricato e' parte del programma.
In altre parole l' istruzione ed il valore sono immessi uno dietro l' altro in due adiacenti locazioni di memoria.
Fino ad ora sono stati visti solo esempi del modo di INDIRIZZAMENTO IMMEDIATO. Vediamo ora gli altri metodi.

INDIRIZZAMENTO IMPLICITO

Questo modo, chiamato qualche volta anche indirizzamento inerente, e' probabilmente il piu' facile da usare in quanto e'il 8502 ad eseguire tutto il lavoro.
Con questo modo possono essere utilizzate numerose istruzioni come TYA, TXA, RTS in quanto il 8502 stesso calcola gli indirizzi.
Fondamentalmente le istruzioni possono dividersi in due gruppi separati.

Nel primo gruppo possono essere messe le istruzioni che sono eseguite interamente entro il 8502 come TYA che trasferisce Y in A e quindi tutto avviene all' interno del microprocessore.
Nel secondo gruppo possiamo mettere invece le istruzioni dove e' necessario un riferimento esterno come per esempio RTS.
Le istruzioni del primo gruppo sono:

DEX DEY INX INY TAX TXA TYA CLC CLD CLI CLV NOP SEC SED SEI.

Quelle del secondo gruppo:
RTS BRK PHA PHP PLA PLP RTI

INDIRIZZAMENTO ASSOLUTO

Le istruzioni usate in questo modo sono facili da comprendere in quanto l' operando dell' istruzione (il numero che viene accanto all' istruzione stessa) e' un numero di 2 Bytes che definisce appunto l' indirizzo in modo assoluto. In questo modo, l' istruzione STA 901 comunica al registro A ESATTAMENTE dove immagazzinare il suo contenuto. Le istruzioni che utilizzano questa forma di indirizzamento sono elencate di seguito ed in parte sono gia' state viste mentre altre le vedremo in seguito:

ADC CMP CPX CPY JMP JSR LDA LDX LDY STA STX STY AND EOR ORA SBC

INDIRIZZAMENTO IN PAGINA ZERO

Questa forma di indirizzamento e' in realta' una sotto-forma dell' indirizzamento assoluto solo che l' operando e' ristretto ad un Byte cioe' massimo 256 caratteri. Il maggior vantaggio di

questo tipo di indirizzamento e' la velocita' di esecuzione perche' le istruzioni sono eseguite in soli tre cicli invece che in quattro come nell' indirizzamento assoluto normale.
A causa della maggior velocita' di esecuzione la pagina zero e' adoperata quasi per intero dal Sistema Operativo e dall' interprete BASIC per cui non e' realmente diponibile per l' Assembler. Al momemto si possono utilizzare le locazioni di pagina zero da 251 a 254.
Quando ne saprete di piu' sull' interpete Basic potrete utilizzare altre locazioni di questa pagina ed addirittura spostare la pagina zero con il suo contenuto da altre parti della memoria. Considerazioni piu' approfondite esulano pero' dagli scopi di questo manuale.
Malgrado sia pericoloso utilizzare le locazioni di questa pagina si possono pero' leggere ed utilizzare le informazioni qui contenute.
Tre locazioni utili di questa pagina possono essere la 160, 161 e 162 che contengono il valore del clock o JIFFIES CLOCK o OROLOGIO (espresso in ore, minuti e secondi) che si incrementa di un 1/60 di secondo.

INDIRIZZAMENTO INDICIZZATO ASSOLUTO

In questo modo un indirizzo viene calcolato usando il contenuto di un registro aggiunto ad un dato indirizzo. E' stato usato di frequente per stampare caratteri sullo schermo con la forma STA LOC,X e STA LOC,Y . Nel programma 20 STA LOC,Y era stato usato in questo modo con il comando STA 1024,Y
Quando si usa questo sistema di indirizzamento bisogna fare attenzione perche' il modo di comportarsi del registro X rispetto al registro Y e' diverso. Entrambi i registri possono essere usati con istruzioni di indicizzamento assoluto,

per esempio operando con due Bytes.
I codici mnemonici che devono essere usati in pagina ZERO devono avere l' indirizzo della pagina che sara' costituito da un solo Byte.

INDIRIZZAMENTO RELATIVO

Negli indirizzi relativi, un salto viene definito relativamente all' attuale posizione del programma. Per esempio l' operando che esprime la posizione desiderata.
Tutte le istruzioni di salto usate in questo modo utilizzano l' indirizzamento relativo.
Il gruppo e' composto dai seguenti comandi:

BCC BCS BEQ BMI BNE BPL BVC BVS

INDIRIZZAMENTO INDIRETTO

Questo e' allo stesso tempo il piu' complesso ed il piu' versatile di tutti i modi di indirizzamento. Questo modo prende il nome di INDIRETTO dal fatto che l' operando e' un puntatore e non un indirizzo. Ed e' questo puntatore che dirige il 8502 attraverso le locazioni di memoria che contengono l' indirizzo.
Ancora una volta tuttavia i meccanismi di indicizzazione di X e Y differiscono fra loro in misura considerevole e danno luogo a diversi modi di indirizzamento.
A causa che indirizzano solo un byte possono puntare solo a locazioni in pagina zero e percio' sono sottoposte alle stesse restrizioni gia' viste per gli altri comandi in pagina zero.

USO DEL REGISTRO X

Con un indirizzamento indiretto che usi il registro X, l' operando e' indicizzato (aggiunto) con il contenuto dello stesso registro per produrre il puntatore. Questa locazione e quella immediatamente successiva sono quindi esaminate ed i loro contenuti forniscono gli indirizzi per i data richiesto con l' ordine seguente:

Byte meno significativo (LSB)

Byte piu' significativo (MSB)

Questa tecnica e' utile per esaminare un particolare elemento in una tavola, essendo fissato l' attuale posizione della tavola dal valore del registro X.
Data la scarsa disponibilita' dello spazio sulla pagina zero del C128 il modo di indirizzamento e' di uso limitato, tuttavia, a scopo dimostrativo, faremo vedere un programma dove viene usata una istruzione di questo tipo.

LDA (LOC),X LoaD A Indirectly indexed with X

Cioe' carica l' Accumulatore con l' indirizzo indiretto indicizzato con il contenuto di X.
Questo tipo di indirizzamento e' conosciuto come INDIRIZZAMENTO INDICIZZATO INDIRETTO o, molto piu' chiaramente INDIRIZZAMENTO PREINDICIZZATO INDIRETTO.
Infatti, come e' implicito nel nome stesso, questo indirizzamento e' preindicizzato poiche' il valore di X e' aggiunto prima che il 8502 salti all' indirizzo.

USO DEL REGISTRO Y

Usando l' indirizzamento indiretto con il registro Y si opera in modo differente, poiche' l' istruzione operando punta direttamente ad una locazione di memoria in pagina zero. Questa contiene il LSB dell' indirizzo e la successiva locazione di memoria contiene il MSB.
Finalmente il contenuto indicizzato del registro e' aggiunto a questo indirizzo per formare l' indirizzo finale indicizzato.
Non deve sorprendere quindi se questa forma e' chiamata anche INDIRIZZO INDIRETTO POSTINDICIZZATO in qunto l' indicizzazione e' calcolata DOPO che l' indirizzo e' stato trovato.
L' interprete Basic ed il Sistema Operativo del C128 fanno un uso molto esteso di questa istruzione. Quando avrete una maggiore confidenza con l' uso dell' Assembler potrete vedere come lavori il Basic e trarne notevole vantaggio nell' uso delle routines del Basic stesso ed in generale del Sistema Operativo del computer.

INDIRIZZAMENTO INDIRETTO ASSOLUTO

Questo modo di indirizzamento e' usato con una sola istruzione:

JMP (LOC) JuMP Indirectly Addressed

Cioe' salta ad un indirizzo indiretto
E' questa un' istruzione assoluta nel quale l' operando e' un indirizzo di 2 bytes e puo' quindi indirizzare una qualsiasi locazione di memoria. E' tuttavia indiretto in quanto a quella locazione ed a quella successiva trova l' indirizzo (prima LSB e poi MSB) per l'

istruzione di salto.
Vediamo un esempio:

```
JMP ($A000)
```

in questo caso eseguira' un salto all' indirizzo di memoria puntata dalle locazioni $A000 e $A001 che in questo caso e' $E394.

LO STACK

L' area Stack (abbreviazione S) e' un blocco di memoria che e' grado di contenere e di manipolare fino a 255 Bytes.
E' usato per un trasferimento di dati che vengono immessi a partire dalla locazione di memoria 511 all' INDIETRO mentre l' indirizzo della prima locazione di memoria libera viene immagazzinata nello STACK POINTER (SP).

NOTA

Fare attenzione perche' nella configurazione 128 dall' indirizzo decimale 256 a 312 di pagina zero sono gia' occupati come vedremo nel manuale di prossima edizione il Sistema Operativo del 128.

IL MONITOR

Il MONITOR e' un programma in linguaggio macchina incorporato che permette di scrivere facilmente programmi in linguaggio macchina. Il MONITOR non e' solo un monitor di linguaggio macchina, un miniassembler e un disassembler.
I programmi in linguaggio macchina scritti utilizzando il MONITOR possono essere utilizzati autonomamente oppure venire usati come subrutine molto veloci per programmi BASIC, dato che il MONITOR puo' tranquillamente coesistere con il BASIC.

COMANDI MONITOR

A ASSEMBLA Assembla una riga del codice 6502.
C CONFRONTA Confronta due sezioni della memoria e segnala le differenze.
D DISASSEMBLA Disassembla una riga del codice 6502.
F RIEMPI Riempie la memoria con il byte specificato.
G ESEGUI Avvia l'esecuzione all'indirizzo specificato.
H CERCA Ricerca nella memoria tutte le posizioni di determinati byte.
L CARICA Carica un file dal nastro o dal disco.
M VISUALIZZA MEMORIA Visualizza i valori esadecimali delle locazioni di memoria.
R VISUALIZZA REGISTRI Visualizza i registri 6502.
S SALVA Salva su nastro o su disco.
T TRASFERISCI Trasferisce il codice da una sezione della memoria all'altra.
V VERIFICA Confronta la memoria con il nastro o con il disco.

X ESCI Uscita da MONITOR
. (punto) Assembla una riga del codice 6502.
> (maggiore di) Modifica la memoria.
; (punto e virgola) Modifica la visualizzazione del Registro 8502.

COME UTILIZZARE IL MONITOR

Si accede al monitor digitando:

MONITOR

il C128 risponde visualizzando i registri 8502 e facendo lampeggiare il cursore. Il cursore rappresenta il prompt che ricorda che il MONITOR e' in attesa dei comandi.

DESCRIZIONE DEI COMANDI

COMANDO: A

SCOPO: introduce una riga del codice che sara' assemblato.

SINTASSI: A <indirizzo> <codice operativo mnemonico> <operando> <inidirizzo>. E' un numero esadecimale che indica la locazione della memoria per il collocamento del codice operativo.

<codice operativo mnemonico>. Mnemonico in linguaggio assembler per tecnologia MOS standard, per es. LDA, STX, ROR, ecc...

<operando>. L'operando, quando richiesto, puo' essere di una qualsiasi delle modalita' di indirizzamento legale. (Per le modalita' di

pagina-zero un numero esadecimale di due cifre e' quello i cui valori sono inferiori a $100. Per indirizzi di pagina-non zero vengono richiesti numeri esadecimali di 4 cifre).
Un RETURN viene utilizzato per indicare la fine della riga di assemblaggio. Se nella riga risultano degli errori, verra' visualizzato un punto di domanda ad indicare un errore e il cursore si spostera' alla riga seguente. Per correggere eventuali errori della riga, potra' essere utilizzato lo screen editor. Una volta assemblata con successo una riga del codice, l'assemblatore stampa un prompt contenente la successiva locazione legale di memoria per una eventuale per una eventuale istruzione, cosi' che A e il numero di riga non dovranno essere battuti piu' di una volta in caso di introduzione nel C 128 di programmi in linguaggio assembler. Vediamo un esempio:

```
.A 3000 LDX # $02
.A 3002
```

NOTA: Un punto (.) equivale al comando ASSEMBLA. Esempio:

```
.1000 LDA # $02
```

COMANDO: C

SCOPO: Confronta due aree di memoria

SINTASSI: C <indirizzo 1> <indirizzo 2> <indirizzo 3>

<indirizzo 1> e' un numero esadecimale indicante l'indirizzo iniziale dell'area di memoria da confrontare.

<indirizzo 2> e' un numero esadecimale indicante l'indirizzo finale dell'area di memoria da confrontare.

<indirizzo 3> e' un numero esadecimale indicante l'indirizzo iniziale dell'altra area di memoria da confrontare.

Se le due aree di memoria sono uguali, MONITOR stampa un RETURN, che indica che la seconda area di memoria e' uguale alla prima. Gli indirizzi dei byte delle due aree che presentano differenze vengono stampati sullo schermo.

COMANDO: D

SCOPO: Disassembla il codice macchina in mnemonici e operandi di linguaggio assembler.

SINTASSI: D [<indirizzo>] [<indirizzo 2>]

<indirizzo> E' un numero esadecimale che imposta l'indirizzo per avviare il disassemblaggio.

<indirizzo 2> E' un indirizzo finale esadecimale opzionale del codice da disassemblare.

Il formato del disassemblato e' solo leggermente diverso dal formato di input di un assemblaggio. La differenza sta nel fatto che il primo carattere di un disassemblato e' una virgola invece di una A (per la leggibilita'), e che l'esadecimale del codice viene anch'esso listato. Un listato di disassemblato puo' essere modificato utilizzando lo screen editor. Operare dei cambiamenti sul mnemonico o sull'operando o sullo schermo e quindi premere RETURN. Cio' introduce la riga e richiama l'assembler per ulteriori modifiche. Un disassemblato puo' essere visualizzato a pagine. L'introduzione di

una D fa si che la pagina successiva del disassemblato venga fatta scorrere sullo schermo.

```
ESEMPIO: D B070   A5 02      * $02
         . B072   20 D2 B8 JSR $B8D2
         . B075   8A          TXA
         . B076   20 D2 FF JSR $FFD2
```

COMANDO: F

SCOPO: Riempie una fascia di locazioni con un byte specificato

SINTASSI: F <indirizzo 1> <indirizzo 2> <byte>

<indirizzo 1> E' la prima locazione da riempire con il <byte>

<indirizzo 2> E' l'ultima locazione da riempire con il <byte>

<valore del byte> E' il numero esadecimale di 1 o 2 cifre che deve essere scritto.

Questo comando e' utile per inizializzare le strutture dei dati o qualsiasi altra area RAM. Vediamo un esempio:

```
F 0400 0518 EA
```

Riempie con $EA (una istruzione NOP) le locazioni di memoria da $0400 a $0518.

COMANDO: G

SCOPO: Iniziare l'esecuzione di un programma all'indirizzo specificato.

SINTASSI: G [<indirizzo>]

<indirizzo> E' un argomento opzionale che specifica il nuovo valore del contatore del programma e fornisce l'indirizzo da cui deve cominciare l'esecuzione. Nel caso in cui <indirizzo> venga tralasciato, l'esecuzione inizia dal CP corrente. (Il CP corrente puo' essere visualizzato utilizzando il comando R.).Il comando GO ripristina tutti i registri (visualizzabili con il comando R) ed inizia l'esecuzione dall'indirizzo iniziale specificato. Si raccomanda cautela nell'uso del comando GO. Per tornare a TEDMON dopo l'esecuzione di un programma in liguaggio macchina, utilizzare l'istruzione BRK.

ESEMPIO: G 1400C

L'esecuzione inizia dalla locazione $140C.

COMANDO: H

SCOPO: Ricerca tutte le posizioni di determinati byte all'interno di una fascia specificata nella memoria.

SINTASSI: H <indirizzo 1> <indirizzo 2> <dati>

<indirizzo 1> indirizzo iniziale della procedura di ricerca

<indirizzo 2> indirizzo finale della procedura di ricerca

<dati> l'insieme di dati per la ricerca dei dati puo' essere una stringa esadecimale o ASCII. Una stringa ASCII viene specificata facendo precedere il primo carattere da una sola

virgoletta, per es. "STRING. I dati possono essere argomenti ad elementi singoli o multipli. Se multipli o in esadecimale, ogni numero deve essere separato da uno spazio.

COMANDO: L

SCOPO: Carica un file da una cassetta o da un disco.

SINTASSI: L <nomefile>,<dispositivo>

<nomefile> e' qualsiasi nome legale di file del C128 tra virgolette.

<dispositivo> e' un numero esadecimale indicante il dispositivo dal quale caricare.

1 cassetta
8 disco (o 09, 0A, ecc...)

Il comando Load fa si che un file venga caricato nella memoria. L'indirizzo iniziale e' contenuto nei primi due byte del file (file di programma). In altre parole, il comando Load carica sempre un file nello stesso posto da cui e' stato salvato. Questo e' molto importante per le operazioni in linguaggio macchina, dato che pochi programmi sono completamente rilocabili. Il file viene caricato nella memoria finche' non si incontra un indicatore di fine file (EOF). Per esempio:

```
L "PIPPO",01   Legge un file dalla cassetta
L "PLUTO",08    Legge un file dal disco
```

COMANDO: M

SCOPO: Visualizza la memoria come una stampa esadecimale e ASCII all'interno della specifica gamma indirizzo.

SINTASSI: M [<indirizzo 1>] [<indirizzo>]

[<indirizzo 1>] E' il primo indirizzo di stampa della memoria.
Opzionale. In caso venga omesso, verr' visualizzata una pagina. Il primo byte e' l'ultimo indirizzo specificato.

[<indirizzo 2>] E' l'ultimo indirizzo di stampa della memoria.
Opzionale. In caso venga omesso, verra' visualizzata una pagina. Il primo byte e' rappresentato dai dati di [<indirizzo 1>].

La memoria viene visualizzata nel seguente formato:

```
>A048 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
```

Il contenuto della memoria puo' essere corretto utilizzando lo screen editor. Spostare il cursore sui dati da modificare, battere la correzione desiderata e premere RETURN. In caso di locazione RAM scorretta o di un tentativo di modifica della ROM, verra' visualizzato un flag(?) d'errore. Una stampa della memoria ASCII dei dati viene visualizzata invertita (per contrastare la stampa degli altri dati visualizzati sullo schermo) alla destra dei dati esadecimali. Quando un carattere non e' stampabile, verra' visualizzato come un punto invertito (.).
Come per il comando di disassemblaggio, si potra' visualizzare la pagina successiva battendo M e RETURN.

COMANDO: >

SCOPO: Puo' essere utilizzato per impostare contmporaneamente da 1 a 8 locazioni di memoria.

SINTASSI: > indirizzo byte di dati 1 <byte di dati da 2 a 8>

indirizzo: primo indirizzo di memoria da impostare
byte di dati 1: dati da collocare all'indirizzo
<byte di dati da 2 a 8>: dati da collocare nelle locazioni di memoria successive al primo indirizzo. Opzionale.

ESEMPIO:

> 1000 09 colloca uno 09 alla locazione 1000
> 3000 23 45 65 colloca un 23 alla locazione 3000, un 45 alla 3001 e un 65 alla 3002

COMANDO: R

SCOPO: Evidenzia importanti registri 8502. Vengono visualizzati il registro dello stato di programma, il contatore di programma, l'accumulatore, i registri di indice X e Y e il puntatore di stack.

SINTASSI: R

ESEMPIO: R

```
        PC    SR  AC  XR  YR  SP
        1002  0F  AA  03  04  FF
```

NOTA: il ; (punto e virgola) puo' essere usato per modificare le visualizzazioni di registro,

allo stesso modo in cui usando > e' possibile modificare i registri di memoria.

COMANDO: S

SCOPO: Salva il contenuto della memoria su nastro o su disco.

SINTASSI: S <"nomefile">,<unita'>, <indirizzo 1>, <indirizzo 2> <"nomefile">. Tutti i nomi file del Plus/4 ammessi. Per salvare i dati, il nome del file deve essere racchiuso tra virgolette. Non e' possibile utilizzare apici.

<unita'> Due possibili dispositivi possono essere cassette e disco. Per memorizzare su cassetta, utilizzare l'unita' 1: il numero del'unita' a disco del Plus/4 e' solitamente 8. Comunque sia, questo puo' essere modificato (per esempio, quando vengono usate piu' unita'). Vedere il MANUALE DELL'UNITA' A DISCO del C128.

<indirizzo 1> indirizzo iniziale di memoria da salvare.

<indirizzo 2> indirizzo finale di memoria da salvare + 1.
Vengono salvati tutti i dati fino al byte di dati di questo indirizzo escluso. Il file creato da quest'o comando e' un file di programma. I primi due byte contengono l'indirizzo base <indirizzo 1> dei dati. Il file potra' essere richiamato utilizzando il comando l.

ESEMPIO: S "PIPPO",8,A400,ABFF

Salva il contenuto della memoria sul disco a partire da $A400 fino a $ABFF

COMANDO: T

SCOPO: Trasferisce segmenti di memoria da un'area di memoria a un'altra.

SINTASSI: T <indirizzo 1> <indirizzo 2> <indirizzo 3>

<indirizzo 1> indirizzo di partenza dei dati da trasferire.

<indirizzo 2. indirizzo finale dei dati da trasferire.

<indirizzo 3> indirizzo di partenza della nuova locazione (dove i dati devono essere trasferiti).

I dati possono essere trasferiti da indirizzi bassi o indirizzi alti o viceversa. Dei segmenti di memoria aggiuntivi di qualsiasi lunghezza possono essere spostati in avanti o all'indietro di un numero qualsiasi di byte (cioe' trasferiti).

ESEMPIO: T 0400 0600 0401

fa scorrere di un byte i dati in memoria da $ 0400 a $ 0600 incluso.

COMANDO: V

SCOPO: Verifica un file su cassetta o su disco confrontandolo con il contenuto della memoria.

SINTASSI: V <"nomefile">,<dispositivo>

<nomefile> e' un qualsiasi nome di file ammesso.

<dispositivo> e' un numero esadecimale che indica in quale unita' si trova il file, la cassetta e' 1 o 01; il disco e' 8 o 08, 09, ecc...

Il comando Verify confronta un file con il contenuto della memoria. Il computer risponde con VERIFYING (verifica in corso). Se viene riscontrato un errore, viene aggiunta la parola ERROR (errore); se la verifica ha buon esito, riappare il cursore lampeggiante.

ESEMPIO: V "PAPERINO", 8

COMANDO: X

SCOPO: Torna al BASIC

SINTASSI: X

Quando viene dato il comando X, il puntatore stack della macchina viene portato al suo valore corrente (vedere il comando R). Se questo fosse modificato dopo essere tornati al BASIC, utilizzare il comando BASIC CLR per re-impostare i puntatori.

COMANDO: @

SCOPO: visualizza lo status del disco

SINTASSI: @ [<n.unita'>], <stringa>

Restituisce lo Status attuale del dischetto. Puo' essere utilizzato anche per inviare un comando.

DESCRIZIONE DELLE ROUTINES KERNAL

Nome della funzione: C64MODE

FUNZIONE:inserisce il modo C64

INDIRIZZO:$FF4D - 65357

DESCRIZIONE :Saltando a questo indirizzo, cioe' chiamando in funzione questa routine si passa dal modo 128, che e' quello di DEFAULT al modo 64. La frequenza viene ridotta ad 1 MHz e la MMU chiude tutti i registri di accesso a quel modo in modo tale che non si possa tornare indietro. Non ci sono parametri di ingresso o di uscita perche' non esiste possibilita' di ritornare indietro.

Nome della funzione: DMA-CALL

FUNZIONE:inizializazione della RAM esterna

INDIRIZZO:$FF50 - 65360

DESCRIZIONE : Per avere un accesso diretto alla memoria verso un' espansione RAM esterna e' necessario per prima cosa chiamare in funzione questa routine.
Nel registro X deve essere indicata la nuova configurazione del sistema per la gestione completa della memoria.

Nome della funzione: BOOTCALL

FUNZIONE:esegue il BOOT da disco

INDIRIZZO:$FF53 - 65363

DESCRIZIONE : Facendo entrare in funzione questa routine si esegue il BOOT del disco. Si carica l' indirizzo che e' nel drive. Accade lo stesso come quando si accende l' apparecchio. Se la routine non trova un BOOTFILE allora il controllo viene restituito all' unita' centrale. Nel registro X e' inserito l' indirizzo della periferica collegata.

Nome della funzione: PHOENIX

FUNZIONE: partenza a freddo

INDIRIZZO:$FF56 - 65366

DESCRIZIONE Partenza a freddo del modo 128. Se viene rilevata la presenza di un cartridge di espansione nella relativa porta allora questa assume il controllo. Dovrebbe esserci l' AUTOSTART. In caso contrario il controllo passa all' unita' a disco. I valori assegnati ai tabulatori, ai tasti funzione ecc. vengono resettati.

Nome della funzione: LKUPLA

FUNZIONE: Ricerca il FILENUMBER

INDIRIZZO: $FF59 - 65369

DESCRIZIONE: Questa routine effettua la ricerca dei parametri di un file basandosi sull' indirizzo logico immagazzinato nell' Accumulatore. LKUPLA esegue iun CLEAR della variabile di STATUS ed in rapporto ai risultati che ottiene dall' esame del registro restituisce un errore se non si trova un indirizzo logico

(LA) nell' accumulatore. Oppure immettera' il primo indirizzo nel registro X e l' indirizzo secondario nel registro Y. Deve essere chiamata prima di un OPEN con un JSR.

Nome della funzione: LKUPSA

FUNZIONE: Ricerca i parametri di un file

INDIRIZZO: $FF5C - 65372

DESCRIZIONE: Ricerca i parametri di un file basandosi sul valore dell' indirizzo secondario immagazzinato nel registro Y. Per il resto e' abbastanza simile come comportamento alla routine precedente.

Nome della funzione: SWAPPER

FUNZIONE:passa da 40 a 80 colonne

INDIRIZZO:$FF5F - 65375

DESCRIZIONE Questa routine inverte il modo 40/80 colonne. Inoltre modifica le informazioni in Pagina zero per lo schermo attivo e lo schermo passivo. La memoria che va E0 fino a FA viene scambiata con la memoria da 0A40 fino a 0A5A. Non e' richiesto nessun parametro di input.

Nome della funzione: DLCHR

FUNZIONE:copia il CHARRROM

INDIRIZZO:$FF62 - 65378

DESCRIZIONE Attivando il tasto ASCII-DIN il set

di caratteri viene di nuovo copiato nel VDC-RAM perche' il controllo delle 80 colonne prende le informazioni per i caratteri non da ROM. Puo' essere utili nella grafica perche' qui il set dei caratteri che si trova nel VDC RAM viene sovrascritto. Questa routine si copia nel VDC RAM il set di caratteri che e' stato selezionato con il tasto ASCII-DIN.
Non sono necessari ne parametri di input ne' di OUTPUT.

Nome della funzione: PFKEY

FUNZIONE: Ridefinizione dei tasti funzione

INDIRIZZO:$FF65 - 65381

DESCRIZIONE Con questa routine si possono assegnare i valori ai tasti funzione.
Nell' accumulatore si trova l' indirizzo della Pagina ZERO che fa da puntatore sul testo del TASTO FUNZIONE. Nel registro X si trova il numero del tasto funzione quindi da 1 a 12. Nel registro Y si trova la lunghezza della stringa. Si chiamera' in funzione questa routine che inserisce la stringa nella tabella.

Nome della funzione: SETBANK

FUNZIONE:definisce il banco di memoria per le operazioni disco

INDIRIZZO:$FF68 - 65384

DESCRIZIONE Questa routine deve essere chiamata prima di ogni comando LOAD, SAVE e VERIFY ed anche prima di OPEN. L' indice di configurazione del nome del file viene consegnato nel Registro Y e l' indice di configurazione del banco di

memoria su cui si lavora viene consegnato nell' Accumulatore. Il registro Y viene memorizzto nella pagina zero all' indirizzo $C6 e l' Accumulatore in $C7.

Nome della funzione: GETCONF

FUNZIONE: prende il Byte di configurazione

INDIRIZZO:$FF6B - 65387

DESCRIZIONE Normalmente esiste una tabella di 16 Bytes di configurazione che e' sufficiente per la definizione dell varie configurazioni. Questa tabella si trova nell' indirizzo $F7F0. La routine consegna al Registro X l' indice di configurazione e riceve nell' Accumulatore il Byte di configurazione che normalmente si scrive nel registro di configurazione che si trova a $FF00

Nome della funzione: JSRFAR

FUNZIONE: salto in subroutine su qualsiasi BANK

INDIRIZZO:$FF6E - 65390

DESCRIZIONE Questa routine ha lo scopo di poter saltare in un qualsiasi sottoprogramma in una qualsiasi configurazione.I parametri sono in Pagina Zero dagli indirizzi da $02 fino a $09. Al terminere della routine la vecchia configurazione viene ristabilita.

Nome della funzione: JMPFAR

FUNZIONE: salta in qualsiasi banco

INDIRIZZO:$FF71 - 65393

DESCRIZIONE Anche con questa routine i parametri vengono consegnati alla pagina zero agli indirizzi da $02 a $09. Tuttavia JMPFAR non e' la chiamata di un sottoprogramma, ma solo un salto in qualsiasi BANK. JMPFAR quindi riunisce in se lo switch del byte di configurazione ed il salto. Poiche' qui non avviene un ritorno non ci saranno dei parametri che vengono restituiti.

Nome della funzione: INDFET

FUNZIONE:prende un Byte di qualsiasi Bank

INDIRIZZO:$FF74 - 65396

DESCRIZIONE Questa routine che si trova sopratutto nella Pagina Zero da la possibilita' di leggere un qualsiasi indirizzo della memoria in qualsiasi configurazione senza dover cambiare in modo notevole la configurazione attuale. Per far cio' e' necessario per prima cosa definire in un indirizzo in Pagina Zero il puntatore alla memoria che si vuole leggere.
Nell' Accumulatore viene poi consegnato questo indirizzo della Pagina Zero. Nel registro X viene consegnato l' indice di configurazione e nel registro Y viene consegnato l'offset riguardo al puntatore della pagina zero.

Nome della funzione: INDSTA

FUNZIONE:memorizza l' Accumulatore in qualsiasi Bank.

INDIRIZZO:$FF77 - 65399

DESCRIZIONE Come avviene nella Routine INDFET cosi questa routine carica nella memoria il contenuto dell' Accumulatore in qualsiasi configurazione della memoria. Quindi anche i parametri devono essere consegnati nell' Accumulatore, nel registro X e nel registro Y. Tuttavia il Byte che deve essere memorizzato finisce nell' Accumulatore. L' indirizzo di Pagina Zero nel quale viene memorizzato il puntatore deve essere definito nell' indirizzo $02B9.

Nome della funzione: INDCMP

FUNZIONE: Confronta l' Accumulatore con la memoria di un Bank qualsiasi.

INDIRIZZO:$FF7A - 65402

DESCRIZIONE Questa routine confronta l' Accumulatore con qualsiasi indirizzo di memoria in qualsiasi banco. Come avviene per la routine INDSTA anche qui e' necessario comunicare l' indirizzo del puntatore in Pagina Zero. Cio' viene fatto nell' indirizzo $02C8. Nell' Accumulatore viene consegnato il Byte che deve essere confrontato, nel registro X viene consegnato l' indice di configurazione e nel registro Y l' OFFSET. Dopo che e' stata chiamata questa routine il risultato di questo confronto e cioe' lo STATUS BYTE del processor si trova nell'indirizzo $05.

Nome della funzione: PRIMM

FUNZIONE:inserisce un testo

INDIRIZZO:$FF7D - 65402

DESCRIZIONE Questa routine e' molto comoda perche' e' molto facile da usare. Infatti non deve essere consegnato alcun parametro. Tutti i Bytes che seguono dopo aver chiamato questa routine vengono consegnati al device di output sul BSOUT.
Come segno che siamo arrivati alla fine viene usato un Byte zero. Il programa viene poi continuato subito dopo il Byte zero. L' unico svantaggio di questa routine e' che quando si va a disassemblere il programma questi diventa poco chiaro e confuso da leggere.

Nome della funzione: ACPTR

FUNZIONE:riceve dati dal bus seriale

INDIRIZZO:$FFA5 - 65445

DESCRIZIONE :Questa e' la routine che si usa quando si desidera ricevere informazioni da una periferica attraverso il BUS seriale, per esempio da disco. Questa routine riceve un Byte di dati dal Bus usando un HANDSHAKING pieno ed il dato e' riportato in Accumulatore. La routine TALK deve essere chiamata in funzione prima di ordinare alla periferica di inviare dati sul Bus.
Se la periferica in ingresso necessita di un comando secondario, questo deve essere inviato usando la routine TKSA prima di ACPTR. Se ci sono errori saranno riportati nella PAROLA di STATO (STATUS WORD) il cui contenuto potra' essere letto dalla routine READST.

Nome della funzione: CHKIN

FUNZIONE: Apre un canale per INPUT

INDIRIZZO: $FFC6 - 65478

DESCRIZIONE: Un qualsiasi File logico che sia stato aperto per mezzo della routine OPEN puo' essere definito come un canale di Input per mezzo di questa routine. Naturalmente la periferica sul canale deve essere una periferica in input, perche' altrimenti avremo un errore e la CHKIN non avra' effetto. Se si stanno ricevendo dati da una qualsiasi altra parte che non sia la tastiera, questa routine (OPEN) deve essere chiamata prima di usare sia le routine CHRIN che GETIN per l' Input dei dati. Se si desidera usare l' Input da tastiera, e nessun altro canale di input e' aperto, allora la chiamata a questa Routine e alla routine OPEN non e' necessaria. Quando questa routine e' utilizzata con una periferica sul bus Seriale, essa invia automaticamente l' indirizzo di chiamata (e l' indirizzo secondario se questo e' specificato in OPEN) sul Bus.

Nome della funzione: CHKOUT

FUNZIONE:Apre un canale per OUTPUT

INDIRIZZO:$FFC9 - 65481

DESCRIZIONE: Un qualsiasi numero di File logico che sia stato creato dalla routine OPEN puo' essere definito come un canale di OUTPUT. Percio' la periferica deve essere una periferica in OUTPUT cioe' in uscita perche' in caso contrario avremo una segnalazione di errore. Questa routine (CHKOUT) deve essere messa in funzione prima che un qualsiasi dato sia inviato ad una periferica (naturalmente in uscita) a

meno che non si desideri usare lo schermo in funzione di periferica in uscita. Quando e' usata per aprire un canale per una periferica sul Bus seriale, questa nostra routine inviera' automaticamente l' indirizzo di LISTEN specificato dalla routine OPEN e l' indirizzo secondario se esiste.

Nome della funzione: CHRIN

FUNZIONE: Riceve un carattere da un canale di Input

INDIRIZZO: $FFCF - 65487

DESCRIZIONE: Questa routine riceve un byte di dati da un canale gia' selezionato per mezzo della routine CHKIN come canale in INPUT. Se CHKIN non e' stata usata per definire un diverso canale di input allora i dati saranno attesi da tastiera. Il Byte di dati e' caricato in accumulatore ed il canale resta aperto.
L' ingresso da tastiera e' manipolato in maniera particolare. Per prima cosa e' attivato il cursore che lampeggera' fino alla digitazione di un ritorno carrello da tastiera (cioe' fino a quando non sia premuto il RETURN). Tutti i caratteri della linea (max 88) sono immagazzinati nel BASIC INPUT BUFFER. Questi caratteri sono recuperati ad uno ad uno per mezzo di tanti salti a questa routine quanti sono questi caratteri.
Quando vine incontrato il ritorno carrello l' intera linea e' stata manipolata.

Nome della funzione: CHROUT

FUNZIONE: Uscita di un carattere

INDIRIZZO: $FFD2 - 65490

DESCRIZIONE: Questa routine fa uscire un carattere su un canale gia' aperto. E' necessario usare le routines OPEN e CHKOUT per fissare un canale di uscita prima di chiamare questa routine. Nel caso che queste chiamate siano omesse i data saranno inviati alla periferica base in uscita, cioe' la numero 3 il video. I Byte che devono uscire , cioe' che sono in Output sono caricati nell' Accumulatore, viene chiamata la routine CHROUT e succesivamente i dati sono inviati alla periferica selezionata, mentre il canale viene lasciato aperto.

Nome della funzione: CIOUT

FUNZIONE: Trasmette un Byte sul bus seriale

INDIRIZZO:$FFA8 - 65448

DESCRIZIONE: Questa routine e' utilizzata per inviare informazioni a periferiche collegate al bus seriale. Percio' la messa in funzione di questa routine avra' come conseguenza l' immissione di un byte di dati sul bus seriale usando un HANDSHAKING seriale pieno. Prima di chiamare questa routine, deve essere chiamata la routine LISTEN che ordinera' alla periferica sul BUS seriale di tenersi pronta a ricevere i dati. (Se alla periferica necessita un indirizzo secondario questo deve essere inviato attraverso l' utilizzo della routine SECOND che vedremo in seguito) La periferica deve essere in ascolto o sara' generato, attraverso la parola di stato , un errore di fuori tempo (TIMEOUT)

Nome della funzione: CINT

FUNZIONE: Inizializza l' editor di schermo e l' integrato 6567

INDIRIZZO:$FF81 - 65409

DESCRIZIONE: Questa routine abilita l' integrato 6567 (VIDEO CONTROLLER) nel C128 per le normali operazioni. Viene inizializzato anche il KERNAL SCREEN EDITOR
Dovrebbe essere chiamata in funzione da un catridge.

Nome della funzione: CLALL

FUNZIONE: Chiude tutti i Files

INDIRIZZO:$FFE7 - 65511

DESCRIZIONE: Questa routine serve per chiudere tutti i files aperti. Quando entra in funzione questa routine i puntatori della tavola dei file aperti sono resettati, chiudendo cosi' tutti i files.
Anche la routine CLRCHN viene chiamata per resettare tutti i canali di I/O.

Nome della funzione: CLOSE

FUNZIONE: Chiude un file logico

INDIRIZZO:$FFC3 - 65475

DESCRIZIONE: Questa routine e' utilizzata per chiudere un file logico dopo che tutte le operazioni di I/O sullo stesso file sono state eseguite. La routine e' chiamata dopo che l' accumulatore e' stato caricato con il numero di

file logico che deve essere chiuso.
Naturalmnete questo sara' lo stesso numero usato quando il file era stato aperto con OPEN.

Nome della funzione: CLRCHN

FUNZIONE: Pulisci i canali di I/O

INDIRIZZO:$FFCC - 65484

DESCRIZIONE: Questa routine e' utilizzata per eseguire il CLEAR di tutti i canali aperti e ripristinare gli stessi canali ai loro valori originari. La periferica normale di INPUT e' 0 (cioe' la tastiera) , mentre la periferica normale di OUTPUT e' 3 (cioe' il video). Se uno dei canali di comunicazione e' su una porta seriale, viene inviato per prima cosa un segnale di UNTALK per eseguire la pulizia del canale di Input o un segnale di UNLISTEN per la pulizia del canale di Output. Non eseguendo la chiamata a questa routine e quindi lasciando gli ascoltatori (LISTENERS) attivi sul bus seriale, diverse periferiche possono ricevere gli stessi dati dal C128 allo stesso tempo. Un sistema per utilizzare questa particolarita' potrebbe essere quello di mettere la stampante in TALK e il disco in LISTEN per consentire la stampa diretta di un file disco. La routine CLRCHN entra automaticamente in funzione dopo l' esecuzione di CLALL.

Nome della funzione: GETIN

FUNZIONE: Riceve un carattere da periferica.

INDIRIZZO:$FFE4 - 65508

DESCRIZIONE: Se il carattere e' da tastiera,

questa routine prende un carattere dalla coda di tastiera (KEYBOARD QUEUE) e lo riporta nell' Accumulatore come valore ASCII. e il buffer (cioe' la coda di tastiera che e' appunto un buffer) e' vuota allora il carattere caricato nell' Accumulatore sara' zero. I caratteri sono immessi nella coda di tastiera utilizzando sia una parte HARDWARE (INTERRUPT DRIVEN KEYBOARD) sia la routine di scnsione della tastiera SCNKEY. Il buffer puo' contenere al massimo 10 caratteri per cui se e' pieno gli altri carateri che si tentera' di immettere saranno ignorati fino a quando almeno un carattere non sia rimosso dalla coda.
Se il canale invece di essere la tastiera e' l' RS-232 allora viene usato solo il registro A e viene riportato un solo carattere e sara' necessario utilizzare READST per il controllo di validita'.
Se il canale invece e' seriale, cassetta o schermo e' chiamata la routine BASIN.

Nome della funzione: IOBASE

FUNZIONE: Definisce la pagina di memria I/O

INDIRIZZO:$FFF3 - 65523

DESCRIZIONE: Questa routine fissa i registri X e Y all' indirizzo della sezione di memoria che definisce dove sono localizzate le periferiche I/O. Questo indirizzo puo' essere utilizzato come linea di deviazione (OFFSET) per accedere alla memoria disegnata per le periferiche I/O. La linea di deviazione e' il numero di locazioni dall' inizio della pagina sulla quale si desidera che i registri I/O siano immessi.

Nome della funzione: IOINIT

FUNZIONE: Inizializza periferiche I/O

INDIRIZZO:$FF84 - 65412

DESCRIZIONE: Questa routine inizializza tutte le periferiche di I/O e le routines. E' normalmente chiamata come parte di una procedura di inizializzazione di un programma su cartridge

Nome della funzione: LISTEN

FUNZIONE: Invia un comando di ascolto ad una periferica sul Bus seriale

INDIRIZZO:$FFB1 - 65457

DESCRIZIONE: Questa routine ordina ad una periferica sul Bus seriale di ricevere dati.L' Accumulatore deve essere caricato con un numero compreso fra 0 e 31 prima di chiamare questa routine.
LISTEN eseguira' un OR logico sul numero bit per bit per convertirlo in un indirizzo di ascolto e poi trasmettera' questo dato come comando sul bus seriale.
La periferica specificata si mettera' allora in modo di ascolto e sara' pronta per ricevere informazioni.

Nome della funzione: LOAD

FUNZIONE: Carica RAM da una periferica

INDIRIZZO:$FFD5 - 65493

DESCRIZIONE: Questa routine carica Bytes di dati

da una qualsiasi periferica in INPUT direttamente entro la memoria del computer.
Puo' anche essere usata per una operazione di verifica che avviene confrontando i dati presenti sulla periferica con quelli in memoria e lasciando i dati in memoria inalterati.
L' Accumulatore deve essere messo a 0 per un' operazione di LOAD o messo a 1 per un' operazione di verifica.
Se la periferica in Input e' aperta con un' indirizzo secondario di 0, allora sara' ignorata la testata (HEADER) dell' informazione.
In questo caso i registri X e Y devono contenere l' indirizzo di partenza per LOAD.
Se la periferica e' collegata con un ' indirizzo secondario 1 0 2, allora i dati saranno caricati in memoria con partenza dall' indirizzo specificato dalla testata.
Questa routine inoltre riporta l' indirizzo della piu' alta locazione di RAM caricata.
Prima di chiamare questa routine e' necessario chiamare le routines SETLFS e SETNAM.
NOTA
Non si puo' eseguire il LOAD da Tastiera (0), RS-232 (2) o schermo (3).

Nome della funzione: MEMBOT

FUNZIONE: Fissa la parte piu' bassa della memoria

INDIRIZZO:$FF9C - 65436

DESCRIZIONE: Questa routine e' usata per fissare la parte piu' bassa della memoria. Se il bit di Carry dell' Accumulatore e' a 1 quando viene chiamata questa routine, allora un puntatore che indica il Byte piu' bassao della RAM e' riportato in X e Y.

Nome della funzione: MEMTOP

FUNZIONE: Fissa la parte alta della memoria

INDIRIZZO:$FF99 - 65433

DESCRIZIONE: Questa routine e' utilizzata per fissare il punto massimo della memoria RAM. Il funzionamento e' simile alla routine precedente (MEMBOT).
Infatti anche in questo caso quando si utilizza questa Routine con il bit di Carry dell' Accumulatore a 1, il puntatore alla fine della memoria RAM e' caricato nei registri X e Y.
Quando invece la si utilizza con il bit di carry a 0, allora il contenuto dei registri X e Y e' caricato nel puntatore al massimo della memoria.

Nome della funzione: OPEN

FUNZIONE: Apre un file logico

INDIRIZZO:$FFC0 - 65472

DESCRIZIONE: Questa routine e' utilizzata per eseguire la funzione di apertura di un File logico. Non appena il file logico e' stato fissato questi puo' essere utilizzato per operazioni di I/O.
Molte delle Routines del sistema operativo fanno uso di OPEN.
Non sono necessari argomenti o operandi ma prima di utilizzare questa routine sara' necessario metterne in funzione altre due cioe' la SETLFS e la SETNAM.

Nome della funzione: PLOT

FUNZIONE: Legge e fissa la posizione del cursore

INDIRIZZO:$FFF0 - 65520

DESCRIZIONE: Quando si salta a questa routine con il carry dell' accumulatore a 1, allora la posizione attuale del cursore , nelle sue coordinate X e Y sara' caricata nel registri Y e X dove Y sara' il numero di colonna del cursore (da 0 a 79) e X il numero di riga occupato dal cursore (da 0 a 24).Se invece il carry e a 0 allora verranno letti i valori dei registri X e Y e il cursore posizionato a quei valori.

Nome della funzione: RAMTAS

FUNZIONE: Controlla le RAM, fissa aree per buffer nastro e schermo

INDIRIZZO: $FF87 - 65415

DESCRIZIONE: Questa routine e' utilizzata per controllare la memoria RAM e fissare i puntatori della memoria sia in alto che in basso. Esegue anche il clear delle locazioni $0000 fino a $0101 e da $0200 a $03FF. Normalmente questa routine e' chiamata come parte di un processo di inizializzazione di un Cartridge.

Nome della funzione: RDTIM

FUNZIONE: Legge l'orologio in tempo reale

INDIRIZZO: $FFDE - 65502

DESCRIZIONE: Questa routine e' utilizzata per leggere il clock o orologio di sistema. La risoluzione del clock, cioe' il tempo minimo e'

di 1 60mo di secondo.
Il risultato della lettura di questa routine e' di 3 Bytes che sono riportati rispettivamente nell' Accumulatore , nel registro X e Y. Operando con questi tre registri e, come vedremo poi con la routine SETTIM e' possibile leggere e variare il contenuto dell' orologio del sistema.

Nome della funzione: READST

FUNZIONE: Legge lo STATUS WORD

INDIRIZZO: $FFB7 - 65463

DESCRIZIONE: Questa routine riporta lo stato attuale delle periferiche in I/O nell' accumulatore. E' utilizzata di norma dopo ogni colloquio con le periferiche e riporta le informazioni sullo stato delle periferiche stesse o eventuali errori incontrati durante operazioni di I/O.
Nome della funzione: RESTOR

FUNZIONE: Reintegra i vettori di sistema.

INDIRIZZO: $FF8A - 65418

DESCRIZIONE: Questa routine reintegra i valori mancanti dei vettori di tutto il sistema usati sia nelle KERNAL che nel BASIC come routines e come interrupts.

Nome della funzione: SAVE

FUNZIONE: Salva la memoria RAM su periferica

INDIRIZZO: $FFD8 - 65496

DESCRIZIONE: Questa routine e' utilizzata per eseguire l' operazione di SAVE di una parte di memoria. La memoria e' salvata da un' indirizzo indiretto in pagina 0 specificato dall' Accumulatore a un' indirizzo immagazzinato nei registri X e Y.
Sara' quindi inviato ad un File logico su una periferica.
Le routines SETLFS e SETNAM devono essere utilizzate prima di accedere a questa routine.
Tuttavia non e' necessario dare un nome al file che si desidera salvare su cassetta, mentre e' necessario per qualsiasi altra periferica.

Nome della funzione: SCNKEY

FUNZIONE: Esegue la scansione di tastiera

INDIRIZZO: $FF9F - 65439

DESCRIZIONE: Questa routine esegue la scansione (cioe' la lettura) della tastiera e controlla se ci sono tasti premuti.
E' la stessa routine chiamata per mezzo della manipolazione di Interrupt.
Se un tasto e' premuto, allora il suo valore ASCII e' immesso nella coda di tastiera.

Nome della funzione: SECOND

FUNZIONE: Invia un indirizzo secondario per la funzione di ascolto (LISTEN)

INDIRIZZO: $FF93 - 65427

DESCRIZIONE: Questa routine e' utilizzata per inviare un indirizzo secondario ad una periferica in I/O dopo che e' stata effettuata

una chiamata alla routine LISTEN e quindi e' stato ordinato alla periferica di porsi in ascolto. Questa routine non puo' essere usata per inviare un indirizzo secondario dopo un salto alla routine TALK (chiamata). Normalmente un indirizzo secondario e' usato per comunicare il tipo di informazione che si desidera inviare alla periferica.

Nome della funzione: SETLFS

FUNZIONE: Fissa il file logico (in maniera completa)

INDIRIZZO: $FFBA - 65466

DESCRIZIONE: Questa routine fissa il numero di file logico, l' indirizzo della periferica e l' indirizzo secondario per le altre routines. Il numero di file logico e' usato dal sistema come chiave di riferimento alla tavola creata dalla routine OPEN file.
L' indirizzo della periferica puo' essere un numero dell' intervallo da 0 a 31.

Nome della funzione: SETMSG

FUNZIONE: Controllo dei messaggi di sistema in uscita

INDIRIZZO: $FF90 - 65420

DESCRIZIONE: Questa routine controlla la stampa di errore ed i messaggi di controllo delle Kernal routines. Sia la stampa dei messaggi di errore come la stampa dei messaggi di controllo possono essere selezionate, cioe' scelte, fissando l' accumulatore quando viene chiamata

la routine.

Nome della funzione: SETNAM

FUNZIONE: Fissa il nome del file

INDIRIZZO: $FFBD - 65496

DESCRIZIONE: Questa routine e' utilizzata per fissare il nome del file per le routine di OPEN, SAVE e LOAD. L' Accumulatore deve essere caricato con la lunghezza del nome del file. I registri X e Y devono essere caricati con l' indirizzo del nome del file secondo il formato 6510 cioe' prima il byte basso e poi il byte alto. L' indirizzo puo' essere un qualsiasi proponibile indirizzo di memoria del sistema dove sia appunto immagazzinata una stringa di caratteri che e' il nome del file. Se non si desidera nessun nome, allora l' Accumulatore deve esere messo a 0 che rappresentera' un file di lunghezza zero. In questo caso i registri X e Y possono essere fissati ad un qualsiasi indirizzo di memoria.

Nome della funzione: SETTIM

FUNZIONE: Fissa i valori del clock di sistema

INDIRIZZO: $FFDB - 65499

DESCRIZIONE: L' orologio di sistema e' mantenuto da una routine di interrupt che lo aggiorna ogni sessantesimo di secondo (un "JIFFY" o ciclo).
Il sistema di clock occupa 3 Bytes che da una capacita' di contare fino a 5.184.000 cicli (o JIFFY) per un totale di 24 ore dopo di che l' orologio torna a zero.

Nome della funzione: SETTMO

FUNZIONE: Fissa il flag di fuori tempo (TIME-OUT) sul bus IEEE

INDIRIZZO: $FFA2 - 65442

DESCRIZIONE: Questa routine fissa il flag di Fuori tempo per la IEEE. Quando questo flag e' messo a 1 il computer attendera' una risposta da una periferica sulla IEEE per 64 millisecondi. Se la periferica non rispondera' al segnale DAV (cioe' DATA ADDRESS VALID) entro questo tempo allora il CBM64 riconoscera' una condizione di errore ed abbandonera' la sequenza di HANDSHAKE. Quando questa routine e' chiamata ed il bit 7 dell' accumulatore contiene uno 0 allora il TIMEOUT e' abilitato, mentre un 1 nello stesso bit dell' accumulatore lo disabilita.

Nome della funzione: STOP

FUNZIONE: Controlla se il tasto di STOP e' premuto.

INDIRIZZO:$FFE1 - 65505

DESCRIZIONE: Se il tasto di STOP era premuto durante la chiamata alla routine UDTIM, la chiamata a questa routine mette a 1 il flag Z. Per di piu' i canali saranno resettati per mancanza di valori, mentre tutti gli altri flags rimarranno immutati. Se il tasto di STOP non era premuto allora l' Accumulatore conterra' un Byte che rappresenta l' ultima riga della scansione di tastiera . L' utente con questo metodo puo' anche controllare alcuni altri tasti.

Nome della funzione: TALK

FUNZIONE: Comando ad una periferica sul BUS seriale di TALK.

INDIRIZZO:$FFB4 - 65460

DESCRIZIONE: Per utilizzare questa routine per prima cosa l' Accumulatore deve essere caricato con un numero di periferica fra 0 e 31. Quando e' chiamata questa routine, allora viene eseguito un OR logico bit per bit per convertire il numero della periferica in un indirizzo di chiamata. Quindi questi dati saranno trasmessi come comando sul Bus seriale.

Nome della funzione: TKSA

FUNZIONE: Invia un indirizzo secondario ad una periferica dopo la routine TALK

INDIRIZZO:$FF96 - 65430

DESCRIZIONE: Questa routine trasmette un indirizzo secondario ad una periferica in attesa di TALK sul bus seriale. Prima di chiamarla deve esserci un numero fra 0 e 31 nell' accumulatore. La routine invia questo numero come un comando di indirizzo secondario sul bus seriale. TKSA puo' essere messa in funzione dopo la chiamata a TALK mentre non operera' dopo una routine o un comando di LISTEN.

Nome della funzione· UDTIM

FUNZIONE: Incrementa l' orologio del sistema

INDIRIZZO:$FFEA - 65514

DESCRIZIONE: Questa routine incrementa l' orologio del sistema. Normalmente questa routine e' chiamata dalla normale routine KERNAL di interrupt ogni sessantesimo di secondo. Se l' utente si programma da se gli interrupt questa routine DEVE essere chiamata per incrementare il temporizzatore. Per di piu',se il tasto STOP e' funzionante, cioe' non e' stato disabilitato, deve essere chiamata anche la routine di STOP che abbiamo visto prima.

Nome della funzione: UNLSN

FUNZIONE: Invia un comando di UNLISTEN

INDIRIZZO:$FFAE - 65454

DESCRIZIONE: Questa routine ordina a tutti le periferiche sul bus seriale di fermare la ricezione dei dati dal computer
In altre parole chiamando questa routine viene inviato un comando di UNLISTEN sul bus seriale.Cio' naturalmente avra' effetto solo sulle periferiche alle quali era stato in precedenza inviato un comando di LISTEN.

Nome della funzione: UNTLK

FUNZIONE: Invia un comando di UNTALK

INDIRIZZO:$FFAB - 65451

DESCRIZIONE: Come la precedente solo che invia un messaggio di UNTALK. Naturalmente anche in questo caso avremo una disabilitazione delle periferiche dal bus seriale .

Nome della funzione: VECTOR

FUNZIONE: Manipola i vettori su RAM

INDIRIZZO:$FF8D - 65421

DESCRIZIONE: Questa routine manipola tutti i sistemi di indirizzi di salto vettorizzati immagazzinati in RAM. Chiamando questa routine con il bit di carry dell' accumulatore a 1, l' attuale contenuto dei vettori della RAM viene immagazzinato in una lista a cui puntano X e Y. Chiamando invece questa routine con lo stesso bit a 0, una lista dell' utente indirizzata dal contenuto dei registri X e Y e' trasferita nel sistema dei vettori RAM

Nome della funzione:CLRWIN

FUNZIONE: cancella una finestra

INDIRIZZO:$C142 - 49474

DESCRIZIONE Se non e' definita alcuna finestra verra' eseguita una pulizia su tutto lo schermo. Se invece questa e' definita la cancellazione avviene entro i limiti della finestra.

Nome della funzione:CURHOM

FUNZIONE: porta il cursore nella posizione HOME all' interno della finestra

INDIRIZZO:$C150 - 49482

DESCRIZIONE Il Cursore viene posizionato nell' angolo superiore sinistro della finestra. Se non e' definita alcuna finestra allora il cursore si

porta nella parte superiore sinistra dello schermo che e' la posizione 0,0.

Nome della funzione:GETLIN

FUNZIONE: prende una riga di input

INDIRIZZO:$C258 - 49752

DESCRIZIONE Dalla tastiera si prendono tanti Bytes che verranno poi scritti sullo schermo nella posizione attuale del cursore fino a che si preme il tasto RETURN.

Nome della funzione:BSOUT SCNR

FUNZIONE: Output di un Byte sullo schermo attuale.

INDIRIZZO:$C72D - 50989

DESCRIZIONE questa routine e' la continuazione della routine BSOUT di indirizzo FFD2 pero' si risparmia alcune richieste prima di arrivare allo schermo per cui e' piu' veloce.
Il Byte viene consegnato nell' accumulatore e scritto sullo schermo attivo in quel momento nella posizione attuale del cursore.

Nome della funzione:CLQIR

FUNZIONE: cancella QUOTE, INS e RVS MODE
INDIRIZZO:$C77D - 51069

DESCRIZIONE Vengono cancellati i FLAG per le virgolette, gli Insert e Reverse .
Questa routine lavora un po' piu' veloce di

quanto avvenga con BSOUT.

LE ALTRE ROUTINES

Vediamo ora alcune altre routine interesanti

$C854 - 51284 Cursor a destra nella finestra
$C85A - 51290 Cursore in basso nella finestra
$C867 - 51303 Cursore in alto nella finestra
$C875 - 51317 Cursore a sinistra nella finestra
$C880 - 51328 Inserisce il II set di caratteri
$C8BF - 51391 Disabilita il modo reverse
$C8C1 - 51393 Abilita il modo reverse
$C8C7 - 51399 Inserisce il modo sottolineatura
$C8CE - 51406 Disabilita la sottolineatura
$C91B - 51483 Cancella il carattere a sinistra del cursore
$C93D - 51517 Cancella il carattere sotto il cursore
$C94F - 51535 Funzione TABULATORE
$C980 - 51584 Disabilita tutti i tabulatori
$C98E - 51598 Fa suonare il campanello (BELL)
$CA14 - 51732 Il cursore definisce la posizione alto sinistra della finestra
$CA16 - 51734 Il cursore definisce la posizione alto a destra della finestra
$CA24 - 51748 Definisce lo schermo per la finestra
$CA52 - 51794 Cancella la riga attuale
$CA76 - 51830 Cancella dalla posizione del cursore fino al termine della riga.
$CA8B - 51851 Cancella dall' inizio riga fino all' attuale posizione del cursore
$CA9F - 51871 Cancella dall' attuale posizione del cursore fino alla fine dello schermo.
$CABC - 51900 Scroll in alto

$CAF2 - 51954 Inserisci il cursore come BLOCCO
$CAFE - 51966 Inserisci il cursore come LINEA
$CB0B - 51979 Disabilita il Flash del cursore
$CB21 - 52001 Abilita il Flash del cursore
$CB3F - 52031 Reverse dello schermo a 80 caratteri
$CB48 - 52040 Schermo a 80 caratteri in modo normale
$CC27 - 52263 Esegue uno Space al posto del cursore
$CC2F - 52271 Preleva il carattere in Accumulatore e lo mette nella posizione attuale del cursore.
$CC4A - 52298 Sulla posizione attuale del cursore viene visualizzato il carattere che si trova nell' Accumulatore, nel colore del registro X alla colonna Y, senza muovere il cursore.
$CC6A - 52330 Fissa la posizione del cursore
$CD2C - 52524 Passa nei modi 40 e 80 colonne.

POSSIBILI MESSAGGI DI ERRORE

Diamo qui di seguito un elenco dei messaggi di errore insieme ad una serie di cause che possono generarli e cerchiamo di indicare i possibili rimedi.
I seguenti messaggii di errore sono normalmente visualizzati anche in un programma BASIC. Il numero dell'errore puo' essere visualizzato utilizzando la funzione ERR $.

1 TOO MANY FILES

C'e' un limite di massimo 10 files che possono stare aperti.In questo caso si consiglia di guardare attentamente il programma per trovare l'errore.

2 FILE OPEN

E' stato tentato di aprire un file che era gia' stato aperto con un comando OPEN oppure e' stato tentato di aprire un fiile con il numero di un file gia' aperto. Controllare il numero di file logico (il primo parametro nel comando open) per assicurarsi che sia utilizzato un diverso numero per ogni file. Inserire un comando CLOSE se si desidera riaprire lo stesso file per una diversa operazione di input/output .

3 FILE NOT OPEN

Si e' tentato di eseguire un accesso ad un file che non era stato preventivamente aperto con un comando OPEN. L'unico rimedio suggerito e' quindi quello di aprire correttamente il file stesso.

4 FILE NOT FOUND

Il nome del file dato con un comando DLOAD o OPEN non e' stato trovato su quella particolare periferica. Controllare che sull'unita' a dischi o sulla cassetta ci siano rispettivamente o il floppy o il nastro giusto. Controllare che sia stato digitato correttamente il nome controllandone cioe' sia le lettere che gli eventuali spazi. Ricordiamo che mentre sul disco la risposta e' immediata, il nastro verra' fatto scorrere fino a quando non termina o comunque non rileva un segnale di END OF TYPE.

5 DEVICE NOT PRESENT

Durante un comando di input o di output il computer si accorge che la periferica alla quale il comando stesso e' stato indirrizzato non risponde al suo segnale. Controllare per prima cosa che la periferica sia correttamente connessa al sistema e accesa. Controllare anche che sia corretto il comando OPEN o gli eventuali altri comandi e controllarne quindi l'indirizzo. Per un esame piu' approfondito di questa funzione vedi il volume "Le Periferiche Commodore" edizione EVM.

6 NOT INPUT FILE

Si e' tentato di eseguire una lettura su di un file che era stato aperto solo per scriverci in altre parole e' stato tentato di impiegare un comando GET o INPUT senza aver eseguito la corretta apertura del file. Controllare quindi i parametri dei comandi ricordando che la lettura richiede che il terzo parametro del comando OPEN sia 0.

7 NOT OUTPUT FILE

Si e' tentato di eseguire una scrittura su un

file presente su cassetta che era stato aperto per la lettura. Anche in questo caso come nel precedente sara' necessario controllare la correttezza dei parametri. Ricordiamo che il terzo parametro del comando OPEN in questo caso deve essere uguale a 1 per eseguire l'operazione di lettura (o uguale a 2 se si desidera un end of tape).

8 MISSING FILE NAME

In un comando di lettura/scrittura e' stato omesso il nome del file. Ricontrollare la linea del programma relativa.

9 ILLEGAL DEVICE NUMBER

E' stato effettuato un tentativo di utilizzare una periferica impropriamente. Puo' verificarsi di tentare di salvare qualcosa ad esempio sulla periferica 0 per lo schermo oppure di leggere qualcosa della periferica stampante. Puo' trattarsi quindi o di un errore di numero relativo all'indirizzo della periferica oppure di un vero e proprio errore di programmazione.

10 NEXT WITHOUT FOR

E' stato incontrato un NEXT senza che sia preceduto dal relativo FOR. Puo' accadere cioe' come vediamo negli esempi sottoriportati, sia che venga incontrato un NEXT senza il FOR, sia che ci si riferisca con il NEXT ad una variabile diversa da quella indicata dal FOR.

ESEMPIO:

For I uguale 1 to 10 : next : next oppure
For I uguale 1 to 10 : next K
E' necessario quindi riesaminare con un po' piu'

di cura l'intero programma o almeno la serie di istruzioni relative ai FOR.

11 SINTAX

Si puo' avere un errore di sintassi sia in modo immediato cioe' mentre si inseriscono i dati, sia durante l'esecuzione di un programma. E' l'errore piu' comune e puo' essere dovuto a numerose cause come:

1) l'utilizzo di parole non riconosciute dal BASIC;
2) la punteggiatura non corretta
3) parole chiave scritte male;
4) uso di parentesi non corretto
Ecc...
Se il sintax error viene segnalato durante l'esecuzione del programma si consiglia di esaminare con cura la linea dove viene segnalato l'errore e correggerla. Ricordiamo che questo tipo di errori in modo programma viene segnalato al momento che il programma stesso e' eseguito cioe' dopo il RUN e non al momento della digitazione da tastiera.

12 RETURN WHITHOUT GOSUB

E' stato incontrato un comando di return senza che sia stato prima eseguito un comando di GOSUB. Eseguire quindi il comando di GOSUB necessario oppure togliere il return se questo e' in piu' cioe' se non e' necessario nessun GOSUB. L'errore spesso e' causato da salti a sub-rutine calcolati ed e' quindi necessario talvolta correggere la logica di funzionamento del programma stesso. Suggeriamo in fase di prova dei programmi di mettere degli END e degli STOP al termine delle sub rutine.

13 OUT OF DATA

E' stato eseguito un comando READ ma tutti i data presenti nel programma sono gia' stati letti. Ad ogni variabile nel comando READ deve corrispondere un elemento di DATA. Quindi o il programma tenta di leggere piu' DATA di quelli che sono presenti oppure i DATA non sono sufficienti. Il possibile rimedio consiste nell'aggiungere altri elementi ai data oppure di restringere il campo di lettura del comando READ. Se si desidera rileggere i data precedentemente utilizzati ricordarsi di immettere un comando RESTORE. (vedi per questo la sezione relativa al comando restore).
Anche il RETURN (ritorno carrello sul messaggio READY presente sul video causera' questo tipo di errore perche' verra' interpretato come un READ Y.

14 ILLEGAL QUANTITY

Si ha questo tipo di errore quando si esegue o si tenta di eseguire una funzione con valori dati ai parametri fuori degli intevalli previsti per quella funzione. E' necessario quindi rivedere i parametri possibili per ciascuna funzione. Questo errore inoltre puo' essere visualizzato se si tenta di far eseguire al programma una funzione USR senza che prima che siano immagazzinati gli indirizzi delle sub-rutine nelle locazioni di memoria 1 e 2. Poiche' il valore del parametro puo' derivare anche da un calcolo eseguire un controllo nel programma per vedere che il valore stesso sia sempre nell'intervallo consentito per quel tipo di funzione.

15 OVERFLOW

In questo caso il calcolo ha dato un risultato maggiore del numero piu' grande manipolabile dal BASIC. Il numero pui' grande permesso e' 1.701411834 E + 38. Eseguire anche in questo caso un controllo del programma. Benche' sia difficile cadere in quest'errore e' possibile correggerlo cambiando l'ordine di esecuzione dei calcoli. N.B. Non esiste un errore di UNDERRFLOW ma un numero minore di 2.93873587 E - 39 non viene distinto dallo 0.

16 OUT OF MEMORY

Questo messaggio puo' apparire non solo mentre sta girando ma anche mentre si sta inserindo o listando un programma. In pratica dice che tutta la memoria a disposizione dell'utente e' stata utilizzata. Inoltre mentre gira il programma ricordiamo che la definizione o la creazione di variabili occupano altra memoria. E' da notare anche che grossissime parti di memoria sono occupate da dimensionamento di matrici anche in presenza di programmi cortissimi. Questo messaggio puo' anche essere causato da un eccessivo uso di cicli di FOR ...NEXT oppure anche di GOSUB che riempiono completamente l'area di memoria dello start. In quest'ultimo caso e' sufficiente eseguire un:

```
? free(0).
```

Se il numero di bytes a disposizione e' nacora abbastanza alto allora e' proprio il caso di ridurre il numero di comandi FOR ...NEXT o di GOSUB. Nei casi precedenti e' bene invece rivedere il programma cercando di ottimizzare gli spazi, controllando accuratamente l'area di dimensionamento delle matrici o spezzando ulteriormente il programma. Per ultimo ricordiamo che una sub-rutine che termina con un

GOTO invece che con un return, puo' causare questo tipo di errore, sempre dovuto all'occupazione dell'area di STACK. E' da notare che questo problema si puo' porre nell'impiego del modo 64 difficilmente verra' fuori nel modo 128 se si sapranno utilizzare accortamente le tecniche di utilizzo dei banchi di memoria.

17 UNDEFINED STATEMENT

Abbiamo tentato di riferirsi ad un numero di linea di programma che in effetti non esiste. Si tratta quindi di un errore di metodologia. Ricontrollare quello che vogliamo fare.

18 BAD SUBSCRIPT

E' stato eseguito un tentativo di riferirsi ad un elemento di una matrice benche' fuori della dimensione della matrice stessa. Questo puo' essere successo perche' e' stata dimensionata (con il comando DIM) una matrice puo' piccola di quella che il programma avrebbe utilizzato, oppure si e' tentato di utilizzare un elemento di indice maggiore di 10 senza aver eseguito il dimensionamento. Si ricorda infatti che le matrici non dimensionate assumono automaticamente indice 10 cioe' da 0 a 10.

19 REDIM'D ARRAY

Il numero di una matrice appare in piu' di un comando DIM. In pratica si e' tentato di dimensionare piu' volte la stessa matrice. Oppure si assegna il valore ad una variabile e poi si tenta di dimensionarla. Esempio:

```
10 A (5) = 21
20 DIM A(10,10)
```

Dopo aver eseguito il RUN avremo un:

? redimed array error

E' necessario solo un po' di attenzione per prevenire questo tipo di errore e eventualmente per correggerlo. Immettere i comandi DIM vicino all'inizio del programma per abitudine in modo di averli rapidamente sotto controllo. Controllare per vedere che ogni comando DIM sia eseguito solo una volta. Ricordare inoltre che un comando DIM non deve essere inserito in un ciclo di FOR NEXT o in una sub-rutine dove possa essere eseguito piu' di una volta.

20 DIVISION BY ZERO

E' stato tentato di eseguire una divisione con divisore uguale a zero cosa che non e' consentita. Controllare il valore delle variabili o anche delle costanti nel numero della linea indicato. Eseguire i necessari cambiamenti nel programma in modo tale che non si trovi a dover eseguire una operazione che dia un risultato di questo genere.

21 ILLEGAL DIRECT

E' stato dato un comando in modo immediato invece puo' essere dato solo in modo programma. I seguenti comandi devono essere dati solo in modo programma:

DATA
DES FN
GET
GET #
INPUT
INPUT #

22 TYPE MISMATCH

Si e' tentato di assegnare valori numerici ad una variabile stringa o viceversa. Oppure ad una funzione che aveva per parametri dei numeri si e' tentato di assegnare una stringa o viceversa.
Esempio:

```
A$ = 5
?Type missmatch error
 Oppure: A ="ciao"
?Type mismatch
```

23 STRING TOO LONG

E' stato tentato di concatenare (ricordiamo con l'operatore piu') 2 o piu' stringhe per creare una nuova stringa piu' grande di 255 caratteri. Spezzare quindi la stringa in 2 o piu' parti di dimensioni minori. Si consiglia inoltre di usare la funzione LEN per calcolare la lunghezza delle stringhe prima di procedere alla concatenazione per evitare questo errore in maniera particolare se la concatenazione delle stringhe, se la loro somma o unione deriva da dati di input.

24 FILE DATA

Sono stati letti dei dati non buoni dal nastro o dal disco. Questo errore puo' derivare non solo da difetti di programmazione ma anche da guasti hardware.

25 FORMULA TOO COMPLEX

Questo non e' un errore vero e proprio ma indica che una espressione del programma stesso, e' troppo complessa o intricata perche' possa essere manipolata dal BASIC. Si consiglia di spezzare l'espressione stessa in due o piu' parti e di far girare il programma, cosi'

corretto fino dall'inizio.

26 CAN'T CONTINUE

E' stato dato un comando di CONT, ma l'esecuzione del programma non puo' riprendere. I motivi possono essere diversi. Il programma e' stato cambiato, e' stato aggiunto qualcosa, o e' stato eseguito un clear in modo diretto. Oppure l'esecuzione del programma e' stata interrotta da un errore. Infatti l'esecuzione non puo' riprendere, nel senso di continuare dallo stesso punto, dopo un messaggio di errore. Eseguire quindi la correzione dell'errore. La strada migliore e' quella di far partire il programma di nuovo da capo con un RUN, tuttavia si puo' tentare di far riprendere l'esecuzione di un programma da un punto determinato tramite l'istruzione di GOTO diretta se non si vogliano perdere dati importanti calcolati sino a quel punto.

27 UNDEF'D FUNCTION

Si e' tentato di utilizzare la funzione definita dall'utente senza che sia stato usato preventivamente una istruzione DES SR per definire la funzione stessa. In questo caso l'unica correzione da effettuare e' quella di definire la funzione prima che vengano eseguiti i calcoli.

28 VERIFY

Questo errore e' la conseguenza dell'esecuzione del programma VERIFY. In questo caso il programma presente in memoria e' stato confrontato con quello su nastro o su disco e non e' stato trovato uguale.

29 LOAD

In questo caso abbiamo un problema di caricamento di programma da periferica sia essa disco o nastro. Si consiglia di riprovare ancora, eventualmente ricorrendo anche al reset del sistema oppure spegnendo la macchina.

30 BREAK

Questo messaggio che non e' un vero e proprio errore, viene visualizzato quando so preme il tasto di RUN-STOP per fermare l'esecuzione del programma.

31 CAN'T RESUME

E' stato incontrato un comando di RESUME senza che sia stato inserito un comando TRAP. Correggere quindi quel punto particolare del programma segnalato dall'errore stesso.

32 LOOP NOT FOUND

Il programma ha incontrato un comando DO e non ha trovato il corrispondente LOOP. Per una comprensione migliore di questo tipo di errore e anche del seguente vedere quanto detta a proposito di FOR NEXT.

33 LOOP WITHOUT DO

E' stato incontrato un comando LOOP senza che sia stato attivato un comando DO. Questo errore si puo' definire simile e opposto al precedente.

34 DIRECT MODE ONLY

Questo errore si verifica quando si e' tentato di inserire in un programma un comando che e'

possibile utilizzare solo in modo diretto.

35 NO GRAPHICS AREA

Prima di utilizzare i comandi tipici della grafica come DRAW, BOX,ecc..., e' necessario inserire il modo grafico, cioe' digitare a livello di programma il comando graphic. Qualora si abbia questa segnalazione di errore evidentemente il comando graphic non e' stato eseguito. Cio' vuol dire che puo' non essere stato inserito nel programma oppure puo' essere messo in un punto errato.

36 BAD DISCK

E' conseguenza di un comando HEADER che e' stato tentato o su un dischetto difettoso o non formattato. Vedi anche il capitolo sulle periferiche.

37 BEND NOT FOUND

Il programma ha incontrato un ciclo di "IF...THEN BEGIN" o "IF...THEN...ELSE BEGIN" e non e' riuscito a trovare la parola chiave THEN per iniziare l'esecuzione di questo ciclo. Si raccomanda di rivedere accuratamente sia la logica di costruzione per vedere se e' necessario il ciclo, sia la corretta disposizione dei comandi all'interno del programma.

38 LINE # TOO LARGE

Il Basic di questo computer nella versione 128 dispone del comando di RENUMBER cioe' di rinumerazione di linee del programm BASIC. Quando viene segnalato questo errore evidentemente e' stato trovato durante la fase

di rinumerazione. In altre parole i parametri di rinumerazione hanno generato o tentato di fornire comunque un programma il cui ultimo numero di linea e' maggiore di 63999 e che per questo tipo 'di BASIC costituisce un errore. Ricordiamo che il processo di rinumerazione in questo caso non e' stato eseguito.

39 UNRESOLVED REFERENCES

Anche questo tipo di errore viene evidenziato durante la rinumerazione del programma BASIC come il precedente. In questo caso un numero di linea alla quale si riferisce un comando ad esempio GOSUB 9000 non esiste. Come sopra la rinumerazione del programma non viene eseguita. In ambedue i casi ricontrollare e diminuire i valori di rinumerazione.

40 UNIMPLEMENTED COMMAND

Evidentemente e' stato trovato un comando che non e' valido per questo tipo di BASIC.

41 FILE READ

E' un errore di lettura di programma che viene fuori quando sta caricando un programma UNSILE da disco. Questo tipo di errore puo' essere dato sia da difetti software che hardware.

PAROLE CHIAVE	ABBREVIAZIONI	PAROLE CHIAVE	ABBREVIAZIONI
ABS	A shift B	DVERIFY	D shift V
APPEND	A shift P	EL	nessuna
ASC	A shift S	END	nessuna
ATN	A shift T	ENVELOPE	E shift N
AUTO	A shift U	ER	nessuna
BACKUP	BA shift C	ERR$	E shift R
BANK	B shift A	EXIT	EX shift I
BEGIN	B shift E	EXP	E shift X
BEND	BE shift N	FAST	nessuna
BLOAD	B shift L	FETCH	F shift E
BOOT	B shift O	FILTER	F shift I
BOX	nessuna	FOR	F shift O
BSAVE	B shift S	FRE	F shift R
BUMP	B shift U	FNXX	nessuna
CATALOG	C shift A	GET	G shift E
CHAR	CH shift A	GETKEY	GETK shift E
CHR$	C shift H	GET#	nessuna
CIRCLE	C shift I	GOSUB	GO shift S
CLOSE	CL shift O	GO64	nessuna
CLR	C shift L	GOTO	G shift O
CMD	C shift M	GRAPHIC	G shift R
COLLECT	COLL shift E	GSHAPE	G shift S
COLINT	nessuna	HEADER	HE shift A
COLLISION	COL shift L	HELP	
COLOR	COL shift O	HEX$	H shift E
CONCAT	C shift O	IF...GOTO	nessuna
CONT	nessuna	IF...THEN...ELSE	nessuna
COPY	CO shift P	INPUT	nessuna
COS	nessuna	INPUT #	I shift N
DATA	D shift A	INSTR	IN shift S
DEC	nessuna	INT	nessuna
DCLEAR	DCL shift E	JOY	J shift O
DCLOSE	D shift C	KEY	K shift E
DEF FN	nessuna	LEFT$	LE shift F
DELETE	DE shift L	LEN	nessuna
DIM	D shift I	LET	L shift E
DIRECTORY	DI shift R	LIST	L shift I
DLOAD	D shift L	LOAD	L shift O
DO	nessuna	LOCATE	LO shift C
DOPEN	D shift O	LOG	nessuna
DRAW	D shift R	LOOP	LO shift O
DSAVE	D shift S	MID$	M shift I

PAROLE CHIAVE	ABBREVIAZIONI	PAROLE CHIAVE	ABBREVIAZIONI
MONITOR	MO shift N	SAVE	S shift A
MOVESHAPE	nessuna	SCALE	SC shift A
MOVSPR	M shift O	SCNCLR	S shift C
NEW	nessuna	SCRATCH	SC shift R
NEXT	N shift E	SGN	S shift G
ON...GOSUB	ON...GO shift S	SIN	S shift I
ON...GOTO	ON...G shift O	SLEEP	S shift L
OPEN	O shift P	SLOW	nessuna
PAINT	P shift A	SOUND	S shift O
PEEK	PE shift E	SPC(	nessuna
PEN	P shift E	SPRCOLOR	SPR shift C
PI	nessuna	SPREDEF	SPR shift D
PLAY	P shift L	SPRITE	S shift P
POKE	PO shift K	SPRSAV	SPR shift S
POS	nessuna	SQR	S shift Q
POT	P shift O	SSHAPE	S shift S
PRINT	?	STASH	S shift T
PRINT#	P shift R	STATUS	nessuna
PRINT USING	?US shift I	STEP	ST shift E
PUDEF	P shift U	STOP	ST shift O
RBUMP	RB shift U	STR$	ST shift R
RCLR	R shift C	SWAP	S shift W
RDOT	R shift D	SYS	nessuna
READ	RE shift A	TAB(	T shift A
RECORD	R shift E	TAN	nessuna
REM	nessuna	TEMPO	T shift E
RENAME	RE shift N	TI	nessuna
RENUMBER	REN shift U	TI$	nessuna
RESTORE	RE shift S	TO	nessuna
RESUME	RES shift U	TRAP	T shift R
RETURN	RE shift T	TROFF	TRO shift F
RGR	R shift G	TRON	TR shift O
RIGHT$	R shift I	UNTIL	U shift N
RLUM	nessuna	USR	U shift S
RND	R shift N	VAL	nessuna
RREG	R shift R	VERIFY	V shift E
RSPCOLOR	RSP shift C	VOL	V shift O
RSPPOS	R shift S	WAIT	W shift A
RSPR	nessuna	WHILE	W shift H
RSPRITE	RSP shift R	WIDTH	WI shift D
RUN	R shift U	WINDOW	W shift I
RWINDOW	R shift W	XOR	X shift O

COMANDI	TOKEN
END	$80
FOR	$81
NEXT	$82
DATA	$83
INPUT#	$84
INPUT	$85
DIM	$86
READ	$87
LET	$88
GOTO	$89
RUN	$8A
IF	$8B
RESTORE	$8C
GOSUB	$8D
RETURN	$8E
REM	$8F
STOP	$90
ON	$91
WAIT	$92
LOAD	$93
SAVE	$94
VERIFY	$95
DEF	$96
POKE	$97
PRINT#	$98
PRINT	$99
CONT	$9A
LIST	$9B
CLR	$9C
CMD	$9D
SYS	$9E
OPEN	$9F
CLOSE	$A0
GET	$A1
NEW	$A2
TAB(	$A3
TO	$A4
FN	$A5
SPC(	$A6
THEN	$A7
NOT	$A8
STEP	$A9
+	$AA
-	$AB
#	$AC
/	$AD
'	$AE
AND	$AF
OR	$B0
>	$B1
=	$B2
<	$B3
SGN	$B4
INT	$B5
ABS	$B6
USR	$B7
FRE	$B8
POS	$B9
SQR	$BA
RND	$BB
LOG	$BC
EXP	$BD
COS	$BE
SIN	$BF
TAN	$C0
ATN	$C1
PEEK	$C2
LEN	$C3
STR$	$C4
VAL	$C5
ASC	$C6
CHR$	$C7
LEFT$	$C8
RIGHT$	$C9
MID$	$CA
GO	$CB
RGR	$CC
RCLR	$CD
POT	$CE $02
BUMP	$CE $03
PEN	$CE $04
RSPPOS	$CE $05
RSPRITE	$CE $06
RSPCOLOR	$CE $07

COMANDI	TOKEN
BACKUP	$F6
DELETE	$F7
RENUMBER	$F8
KEY	$F9
MONITOR	$FA
USING	$FB
UNTIL	$FC
WHILE	$FD
BANK	$FE $02
FILTER	$FE $03
PLAY	$FE $04
TEMPO	$FE $05
MOVSPR	$FE $06
SPRITE	$FE $07
SPRCOLOR	$FE $08
RREG	$FE $09
ENVELOPE	$FE $0A
SLEEP	$FE $0B
CATALOG	$FE $0C
DOPEN	$FE $0D
APPEND	$FE $0E
DCLOSE	$FE $0F
BSAVE	$FE $10
BLOAD	$FE $11
RECORD	$FE $12
CONCAT	$FE $13
DVERIFY	$FE $14
DCLEAR	$FE $15
SPRSAV	$FE $16
COLLISION	$FE $17
BEGIN	$FE $18
BEND	$FE $19
WINDOW	$FE $1A
BOOT	$FE $1B
WIDTH	$FE $1C
SPRDEF	$FE $1D
QUIT	$FE $1E
STASH	$FE $1F
FETCH	$FE $21
STASH	$FE $23
OFF	$FE $24
FAST	$FE $25
SLOW	$FE $26

COMANDI	TOKEN
XOR	$CE $08
RWINDOW	$CE $09
POINTER	$CE $0A
JOY	$CF
RDOT	$D0
DEC	$D1
HEX$	$D2
ERR$	$D3
INSTR	$D4
ELSE	$D5
RESUME	$D6
TRAP	$D7
TRON	$D8
TROFF	$D9
SOUND	$DA
VOL	$DB
AUTO	$DC
PUDEF	$DD
GRAPHIC	$DE
PAINT	$DF
CHAR	$E0
BOX	$E1
CIRCLE	$E2
GSHAPE	$E3
SSHAPE	$E4
DRAW	$E5
LOCATE	$E6
COLOR	$E7
SCNCLR	$E8
SCALE	$E9
HELP	$EA
DO	$EB
LOOP	$EC
EXIT	$ED
DIRECTORY	$EE
DSAVE	$EF
DLOAD	$F0
HEADER	$F1
SCRATCH	$F2
COLLECT	$F3
COPY	$F4
RENAME	$F5

INDICE

Introduzione pag. 5

CAPITOLO PRIMO

Inizializzazione " 9
Modi operativi " 10
Formato della linea " 10
Numeri di linea " 11
La programmazione " 11
Algoritmo informatico " 12
Le righe di un programma " 15

I COMANDI " 18

CAPITOLO SECONDO

Il Basic " 183
Costanti e variabili " 185
Costanti stringa " 188
Le variabili " 189
Espressioni ed operatori " 190
Ordine gerarchico " 193
Operazioni su stringhe " 194

CAPITOLO TERZO

La grafica " 197
I comandi grafici " 198
Modo testo standard " 200
Modo alta risoluzione " 201
Modo multicolore " 202
Modo split screen " 203

Selezione colori " 203
Codici colori schermo 40 col " 204
Codice colori schermo 80 col " 205
Visualizzazione disegni sch. " 206
Le figure ed'i disegni " 208
Visualizzazione caratteri " 210
Gli sprites " 212
Utilizzo dei comandi sprites " 213
Caricamento sprite " 215
Attivazione sprites " 216
Come muovere gli sprites " 218
Gestione della collisione " 221
Modo sprite designer " 225
Sommario degli sprites " 226
Procedura creazione " 227
La manipolazione degli sprites " 231
Immagazzinamento sprites " 232

CAPITOLO QUARTO

Programmazione suoni/musica " 237
Il comando SOUND " 238
La frequenza e le onde sonore " 240
Suoni casuali " 245
Le caratteristiche del suono " 246
Il generatore di inviluppo " 250
Il volume " 252
Il tempo " 253
Il comando PLAY " 253
Il filtro " 256

CAPITOLO QUINTO

Le periferiche " 259
Concetto di file " 261
File programmi " 262
Comandi per i files " 264
VERIFY " 265

SAVE " 266
Data files " 267
RECORD e FIELDS " 267
Files logici e unita' fisiche " 268
Indirizzo secondario " 272
Fisical Unit Status " 274
Manipolazione dati su cassetta " 275
Il formato files su cassetta " 277
Lettura dati da cassetta " 286
OPEN " 288
Chiusura di un file " 290
Come accedere ai data files " 292
IL DISCO " 293
Immagazzinamento dei dati " 294
Directory e BAM " 294
Files relatives " 296
Files sequenziali " 296
Indirizzamento del disco " 296
Preparazione di un disco " 297
Inizializzazione " 299
VALIDATE " 300
RENAME " 301
SCRATCH " 301
COPY " 302
DIRECTORY " 303
COLLECT " 303
DUPLICATE " 303
BACKUP " 304
CONCAT " 305
Manipolazione dei dati disco " 305
SAVE E DSAVE " 306
LOAD e DLOAD " 307
VERIFY " 308
OPEN " 308
DOPEN " 309
CLOSE e DCLOSE " 309
PRINT# " 310
INPUT # " 312
GET # " 313
RECORD # " 314

CAPITOLO SESTO

Il linguaggio macchina e SO	"	315
Il microprocessore	"	317
I sistemi numerici	"	319
L' esadecimale	"	321
Codice ASCII	"	323
L' Accumulatore	"	325
I registri indice	"	326
I salti ed il PC	"	327
Salti incondizionati	"	327
Salti condizionati	"	328
Indirizzamento immediato	"	330
Indirizzamento implicito	"	330
Indirizzamento assoluto	"	331
Indirizzamento in pagina ZERO	"	331
Indirizzamento ind. assoluto	"	332
Indirizzamento relativo	"	333
Indirizzamento indiretto	"	333
Uso dei registri X e Y	"	334
Lo STACK	"	336
Il MONITOR	"	337
Come utilizzare il MONITOR	"	338
Descrizione dei comandi	"	338
Le routines KERNAL	"	349
Le altre routines	"	376
Messaggi di errore	"	378
PAROLE CHIAVE	"	391
TOKEN	"	393

GUIDA DI RIFERIMENTO AL COMMODORE 128

Questa GUIDA e´ il primo di una serie di manuali per il nuovo computer COMMODORE 128.
Inizia con un approfondito esame di TUTTI i comandi con numerosi esempi applicativi.
Segue un´ approfondita trattazione delle grandi capacita´ grafiche e del sintetizzatore sonoro oltre a pratiche dimostrazioni di quanto questi argomenti, di solito difficili, si possano con estrema semplicita´ usare sul C 128.
Sono state dedicate anche numerose pagine alle periferiche collegabili, cassetta e floppy disk, ed al trattamento dei files perche´ il C128 NON E´ un VIDEOGIOCO.
Al termine TUTTO cio´ che e´ necessario conoscere sul Linguaggio Macchina compresi i comandi Assembler e le routines del Sistema Operativo pubblicate PER LA PRIMA VOLTA.
Questa GUIDA e´ la prima e piu´ naturale integrazione al manuale di accompagnamento del C128 e permette di scoprirne i numerosi segreti e le capacita´ operative.

L. 28.000

EVM COMPUTERS s.r.l.
Via Marconi, 9/A - Loc. Muraccio - Tel. (055) 980.242 - 982.513
52025 MONTEVARCHI (Arezzo)



Guida di riferimento al C128